Exhibit II

Exhibit E-7

Invalidity of U.S. Patent No. 7,725,253 ("'253 Patent") under Pre-AIA Section 102 or Section 103 in view of Welch et al., "SCAAT: Incremental Tracking with Incomplete Information," Univ. of N.C. at Chapel Hill, 1997 ("Welch 1997")[1]

Welch 1997 was published in 1997. Plaintiffs belatedly asserted a priority date of June 13, 2001 for the '253 Patent on December 22, 2021, 71 days after the Court's deadline. Defendants have reviewed Plaintiffs' alleged evidence of the purported June 13, 2001 priority date, and maintain that the '253 Patent is not entitled to this priority date. *See* Defendants' March 15, 2022 Supplemental Invalidity Contentions. Defendants reserve their objections to Plaintiffs' belated assertion of the new priority date and expressly reserve all rights to challenge this alleged new priority date. Defendants reserve their objections to Plaintiffs' belated assertion of the new priority date and expressly reserve all rights to challenge this alleged new priority date. As such, Defendants assume for the sake of these invalidity contentions, that the priority date for the '253 Patent is August 9, 2002 based on the first filed Provisional Application from which the '253 Patent claims priority. (Defendants do not concede nor agree that Plaintiffs are even entitled to this date.) Assuming this priority date, Welch 1997 qualifies as prior art under at least pre-AIA Sections 102(a) and (b) to the '253 Patent.

As described herein, the asserted claims of the '253 Patent are invalid (a) under one or more sections of 35 U.S.C. § 102 as anticipated expressly or inherently by Welch 1997 (including the documents incorporated into Welch 1997 by reference) and (b) under 35 U.S.C. § 103 as obvious in view of Welch 1997 standing alone and, additionally, in combination with the knowledge of one of ordinary skill in the art, and/or other prior art, including but not limited to the prior art identified in Defendants' Invalidity Contentions and the prior art described in the claim charts attached in Exhibits E-1 – E-23. With respect to the proposed modifications to Welch 1997, as of the priority date of the '253 Patent, such modification would have been obvious to try, an obvious combination of prior art elements according to known methods to yield predictable results, a simple substitution of one known element for another to obtain predictable results, a use of known techniques to improve a similar device or method in the same way, an application of a known technique to a known device or method ready for improvement to yield predictable results, a variation of a known work in one field of endeavor for use in either the same field or a different one based on design incentives or other market forces with variations that are predictable to one of ordinary skill in the art, and/or obvious in view of teachings, suggestions, and motivations in the prior art that would have led one of ordinary skill to modify or combine the prior art references. All cross-references should be understood to include material that is cross-referenced within the cross-reference. Where a particular figure is cited, the

[1] Discovery in this case is ongoing and, accordingly, this invalidity chart is not to be considered final. Defendants have conducted the invalidity analysis herein without having fully undergone claim construction and a *Markman* hearing. By charting the prior art against the claim(s) herein, Defendants are not admitting nor agreeing to Plaintiffs' interpretation of the claims at issue in this case. Additionally, these charts provide representative examples of portions of the charted references that disclose the indicated limitations under Plaintiffs' application of the claims; additional portions of these references other than the representative examples provided herein may also disclose the indicated limitation(s) and Defendants contend that the asserted claim(s) are invalid in light of the charted reference(s) as a whole. Defendants reserve the right to rely on additional citations or sources of evidence that also may be applicable, or that may become applicable in light of claim construction, changes in Plaintiffs' infringement contentions, and/or information obtained during discovery as the case progresses. Further, by submitting these invalidity contentions, Defendants do not waive and hereby expressly reserve their right to raise other invalidity defenses, including but not limited to defenses under Sections 101 and 112. Defendants reserve the right to amend or supplement this claim chart at a later date, including after the Court's order construing disputed claim terms.

citation should be understood to encompass the caption and description of the figure as well as any text relating to or describing the figure. Conversely, where particular text referring to a figure is cited, the citation should be understood to include the figure as well.

A. INDEPENDENT CLAIM 1

CLAIM 1	Welch 1997
[1.pre] A tracking system comprising:	At least under Plaintiffs' apparent infringement theory, Welch 1997 discloses, either expressly or inherently, a method for tracking an object. No party has yet asserted that the preamble is limiting, nor has the Court construed the preamble as limiting. However, to the extent that the preamble is limiting, it is disclosed by Welch 1997. In the alternative, this element would be obvious over Welch 1997 in light of the other references disclosed in Defendants' Invalidity Contentions and/or the knowledge of one of ordinary skill in the art. *See, e.g.*: **We present a promising new mathematical method for tracking a user's pose (position and orientation) for interactive computer graphics. The method, which is applicable to a wide variety of both commercial and experimental systems, improves accuracy by properly assimilating sequential observations, filtering sensor measurements, and by concurrently autocalibrating source and sensor devices. It facilitates user motion prediction, multisensor data fusion, and higher report rates with lower latency than previous methods.** Tracking systems determine the user's pose by measuring signals from low-level hardware sensors. For reasons of physics and economics, most systems make multiple sequential measurements which are then combined to produce a single tracker report. For example, commercial magnetic trackers using the SPASYN (*Space Synchro*) system sequentially measure three magnetic vectors and then combine them mathematically to produce a report of the sensor pose. **Our new approach produces tracker reports as each new low-level sensor measurement is made rather than waiting to form a complete collection of observations.** Because single observations under-constrain the mathematical solution, we refer to our approach as single-constraint-at-a-time or SCAAT tracking. The key is that the single observations provide some information about the user's state, and thus can be used to incrementally improve a previous estimate. We recursively apply this principle, incorporating new sensor data as soon as it is

Exhibit E-7

CLAIM 1	Welch 1997
	measured. With this approach we are able to generate estimates more frequently, with less latency, and with improved accuracy. We present results from both an actual implementation, and from extensive simulations. Welch 1997 at Abstract. **The method we present requires, we believe, a fundamental change in the way people think about estimating a set of unknowns in general, and tracking for virtual environments in particular.** Most of us have the preconceived notion that to estimate a set of unknowns we need as many constraints as there are degrees of freedom at any particular instant in time. What we present instead is a method to constrain the unknowns over time, continually refining an estimate for the solution, a single constraint at a time. Welch 1997 at Section 1. **The SCAAT method employs a Kalman filter (KF) in an unusual fashion. The Kalman filter is a mathematical procedure that provides an efficient computational (recursive) method for the least-squares estimation of a linear system.** It does so in a predictor-corrector fashion, predicting short-term (since the last estimate) changes in the state using a dynamic model, and then correcting them with a measurement and a corresponding measurement model. The extended Kalman filter (EKF) is a variation of the Kalman filter that supports estimation of nonlinear systems, e.g. 3D position and orientation tracking systems. A basic introduction to the Kalman filter can be found in Chapter 1 of [31], while a more complete introductory discussion can be found in [40], which also contains some interesting historical narrative. More extensive references can be found in [7,18,24,28,31,46]. Welch 1997 at Section 3. The Kalman filter has been employed previously for virtual environment tracking estimation and prediction. For example see [2,5,12,14,42], and most recently [32]. In each of these cases however the filter was applied directly and only to the 6D pose estimates delivered by the off-the-shelf tracker. **The SCAAT approach could be applied to either a hybrid system using off-the-shelf and/or custom trackers, or it could be employed by tracker developers to improve the existing systems for the end-user graphics community.** Welch 1997 at Section 3. **In this section we describe the method in a manner that does not imply a specific tracking system. (In section 3.4 we present experimental results of a specific implementation, a SCAAT wide-area optoelectronic tracking system.)** In section 3.1 we describe the method for tracking, and in section 3.2 we describe one possible

Exhibit E-7

CLAIM 1	Welch 1997
	method for concurrent autocalibration. Welch 1997 at Section 3. **3.1 Tracking** **3.1.1 Main Tracker Filter** The use of a Kalman filter requires a mathematical (state-space) model for the dynamics of the process to be estimated, the target motion in this case. While several possible dynamic models and associated state configurations are possible, we have found a simple *position-velocity* model to suffice for the dynamics of our applications. In fact we use this same form of model, with different parameters, for all six of the position and orientation components $(x, y, z, \phi, \theta, \psi)$. Discussion of some other potential models and the associated trade-offs can be found in [7] pp. 415-420. Because our implementation is discrete with inter sample time δt we model the target's dynamic motion with the following linear difference equation: $$\hat{x}(t+\delta t) = A(\delta t)\hat{x}(t) + \hat{w}(\delta t). \quad (2)$$ Welch 1997 at Section 3.1.

Exhibit E-7

CLAIM 1	Welch 1997

In the standard model corresponding to equation (2), the n dimensional Kalman filter *state vector* $\hat{x}(t)$ would completely describe the target position and orientation at any time t. In practice we use a method similar to [2,6] and maintain the complete target orientation externally to the Kalman filter in order to avoid the nonlinearities associated with orientation computations. In the internal state vector $\hat{x}(t)$ we maintain the target position as the Cartesian coordinates (x, y, z), and the *incremental* orientation as small rotations (ϕ, θ, ψ) about the (x, y, z) axis. Externally we maintain the target orientation as the *external quaternion* $\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z))$. (See [9] for discussion of quaternions.) At each filter update step, the incremental orientations (ϕ, θ, ψ) are factored into the external quaternion $\hat{\alpha}$, and then zeroed as shown below. Thus the incremental orientations are linearized for the EKF, centered about zero. We maintain the derivatives of the target position and orientation internally, in the state vector $\hat{x}(t)$. We maintain the angular velocities internally because the angular velocities behave like orthogonal vectors and do not exhibit the nonlinearities of the angles themselves. The target state is then represented by the $n = 12$ element internal state vector

$$\hat{x} = \left[x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \phi\ \theta\ \psi\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}\right]^T \quad (3)$$

and the four-element external orientation quaternion

$$\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z)), \quad (4)$$

where the time designations have been omitted for clarity.

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 1	Welch 1997

The $n \times n$ *state transition matrix* $A(\delta t)$ in (2) projects the state forward from time t to time $t + \delta t$. For our linear model, the matrix implements the relationships

$$\begin{aligned} x(t+\delta t) &= x(t) + \dot{x}(t)\delta t \\ \dot{x}(t+\delta t) &= \dot{x}(t) \end{aligned} \qquad (5)$$

and likewise for the remaining elements of (3).

The $n \times 1$ *process noise vector* $\vec{w}(\delta t)$ in (2) is a normally-distributed zero-mean sequence that represents the uncertainty in the target state over any time interval δt. The corresponding $n \times n$ *process noise covariance matrix* is given by

$$E\{\vec{w}(\delta t)\vec{w}^T(\delta t + \varepsilon)\} = \begin{cases} Q(\delta t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \qquad (6)$$

Because our implementation is discrete with inter sample time δt, we can use the transfer function method illustrated by [7] pp. 221-222 to compute a *sampled* process noise covariance matrix. (Because the associated random processes are presumed to be time stationary, we present the process noise covariance matrix as a function of the inter-sample duration δt only.) The non-zero elements of $Q(\delta t)$ are given by

$$\begin{aligned} Q(\delta t)[i,i] &= \vec{\eta}[i]\frac{(\delta t)^3}{3} \\ Q(\delta t)[i,j] = Q(\delta t)[j,i] &= \vec{\eta}[i]\frac{(\delta t)^2}{2} \\ Q(\delta t)[j,j] &= \vec{\eta}[i](\delta t) \end{aligned} \qquad (7)$$

for each pair

$$(i,j) \in \{(x,\dot{x}), (y,\dot{y}), (z,\dot{z}), (\phi,\dot{\phi}), (\theta,\dot{\theta}), (\psi,\dot{\psi})\}.$$

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 1	Welch 1997

The $\hat{\eta}[i]$ in (7) are the *correlation kernels* of the (assumed constant) noise sources presumed to be driving the dynamic model. We determined a set of values using Powell's method, and then used these in both simulation and our real implementation. The values can be "tuned" for different dynamics, though we have found that the tracker works well over a broad range of values.

The use of a Kalman filter requires not only a dynamic model as described above, but also a *measurement model* for each available type of measurement. The measurement model is used to predict the ideal noise-free response of each sensor and source pair, given the filter's current estimate of the target state as in equations (3) and (4).

It is the nature of the measurement models and indeed the actual sensor measurements that distinguishes a SCAAT Kalman filter from a well-constrained one.

For each sensor *type* σ we define the $m_\sigma \times 1$ *measurement vector* $\hat{z}_\sigma(t)$ and corresponding *measurement function* $\hat{h}_\sigma(\bullet)$ such that

$$\hat{z}_{\sigma,t} = \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t) + \hat{v}_\sigma(t) . \qquad (8)$$

Note that in the "purest" SCAAT implementation $m_\sigma = 1$ and the measurements are incorporated as single scalar values. However if it is not possible or necessary to isolate the measurements, e.g. to perform autocalibration, then multi-dimensional measurements can be incorporated also. Guidelines presented in [47] lead to the following heuristic for choosing the SCAAT Kalman filter measurement elements (constraints):

During each SCAAT Kalman filter measurement update one should observe a single sensor and source pair only.

For example, to incorporate magnetic tracker data as an end-user, $m_\sigma = 7$ for the three position and four orientation (quaternion)

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 1	Welch 1997

elements, while if the manufacturer were to use the SCAAT implementation, $m_\sigma = 3$ for each 3-axis electromagnetic response to a single excitation. For an image-based landmark tracker such as [41] the measurement function would, given estimates of the camera pose and a single landmark location, transform the landmark into camera space and then project it onto the camera image plane. In this case $m_\sigma = 2$ for the 2D image coordinates of the landmark.

The $m_\sigma \times 1$ *measurement noise vector* $\hat{\nu}_\sigma(t)$ in (8) is a normally-distributed zero-mean sequence that represents any random error (e.g. electrical noise) in the measurement. This parameter can be determined from component design specifications, and (or) confirmed by off-line measurement. For our simulations we did both. The corresponding $m_\sigma \times m_\sigma$ *measurement noise covariance matrix* is given by

$$E\{\hat{\nu}_\sigma(t)\hat{\nu}_\sigma^T(t+\varepsilon)\} = \begin{cases} R_\sigma(t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases} . \qquad (9)$$

For each measurement function $\hat{h}_\sigma(\bullet)$ we determine the corresponding Jacobian function

$$H_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t)[i,j] \equiv \frac{\partial}{\partial \hat{x}[j]} \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t)[i] , \qquad (10)$$

where $1 \le i \le m_\sigma$ and $1 \le j \le n$. Finally, we note the use of the standard (Kalman filter) $n \times n$ *error covariance matrix* $P(t)$ which maintains the covariance of the error in the estimated state.

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 1	Welch 1997

3.1.2 Tracking Algorithm

Given an initial state estimate $\hat{x}(0)$ and error covariance estimate $P(0)$, the SCAAT algorithm proceeds similarly to a conventional EKF, cycling through the following steps whenever a discrete measurement $\check{z}_{\sigma,t}$ from some sensor (type σ) and source becomes available at time t:

a. Compute the time δt since the previous estimate.

b. Predict the state and error covariance.

$$\begin{aligned} \hat{x}^- &= A(\delta t)\hat{x}(t-\delta t) \\ P^- &= A(\delta t)P(t-\delta t)A^T(\delta t) + Q(\delta t) \end{aligned} \tag{11}$$

c. Predict the measurement and compute the corresponding Jacobian.

$$\begin{aligned} \hat{z} &= \check{h}_\sigma(\hat{x}^-, \check{b}_t, \vec{c}_t) \\ H &= H_\sigma(\hat{x}^-, \check{b}_t, \check{e}_t) \end{aligned} \tag{12}$$

d. Compute the *Kalman gain*.

$$K = P^- H^T (HP^- H^T + R_\sigma(t))^{-1} \tag{13}$$

e. Compute the *residual* between the actual sensor measurement $\check{z}_{\sigma,t}$ and the predicted measurement from (12).

$$\vec{\Delta z} = \check{z}_{\sigma,t} - \hat{z} \tag{14}$$

f. Correct the predicted tracker state estimate and error covariance from (11).

$$\begin{aligned} \hat{x}(t) &= \hat{x}^- + K\vec{\Delta z} \\ P(t) &= (I - KH)P^- \end{aligned} \tag{15}$$

Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 1	Welch 1997
	g. Update the external orientation of equation (4) per the change indicated by the (ϕ, θ, ψ) elements of the state.* $\Delta\hat{\alpha} = \text{quaternion}(\hat{x}[\phi], \hat{x}[\theta], \hat{x}[\psi])$ $\hat{\alpha} = \hat{\alpha} \otimes \Delta\hat{\alpha}$ (16) h. Zero the orientation elements of the state vector. $\hat{x}[\phi] = \hat{x}[\theta] = \hat{x}[\psi] = 0$ (17) The equations (11)-(17) may seem computationally complex, however they can be performed quite efficiently. The computations can be optimized to eliminate operations on matrix and vector elements that are known to be zero. For example, the elements of the Jacobian H in (12) that correspond to the velocities in the state $\hat{x}(t)$ will always be zero. In addition, the matrix inverted in the computation of K in (13) is of rank m_σ (2×2 for our example in section 3.4) which is smaller for a SCAAT filter than for a corresponding conventional EKF implementation. Finally, the increased data rate allows the use of the *small angle approximations* $\sin(\theta) = \theta$ and $\cos(\theta) = 1$ in $\check{h}_\sigma(\bullet)$ and $H_\sigma(\bullet)$. The total *per estimate* computation time can therefore actually be less than that of a corresponding conventional implementation. (We are able to execute the SCAAT filter computations, with the autocalibration computations discussed in the next section, in approximately 100µs on a 200 MHz PC-compatible computer.) Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 1	Welch 1997
	3.1.3 Discussion The key to the SCAAT method is the number of constraints provided by the measurement vector and measurement function in equation (8). For the 3D-tracking problem being solved, a unique solution requires $C = 6$ non-degenerate constraints to resolve six degrees of freedom. Because individual sensor measurements typically provide less than six constraints, conventional implementations usually construct a complete measurement vector $$\underset{z}{\rightarrow}_t = \left[\underset{z}{\rightarrow}^T_{\sigma_1, t_1} \cdots \underset{z}{\rightarrow}^T_{\sigma_N, t_N}\right]^T$$ from some group of $N \geq C$ individual sensor measurements over time $t_1 \ldots t_N$, and *then* proceed to compute an estimate. Or a particular implementation may operate in a *moving-window* fashion, combining the most recent measurement with the $N-1$ previous measurements, possibly implementing a form of a finite-impulse-response filter. In any case, for such well-constrained systems complete observability is obtained at each step of the filter. Systems that collect measurements sequentially in this way inherently violate the simultaneity assumption, as well as increase the time δt between estimates. Welch 1997 at Section 3.1.3.

Exhibit E-7

CLAIM 1	Welch 1997

In contrast, the SCAAT method blends individual measurements that each provide incomplete constraints into a complete state estimate. The EKF inherently provides the means for this blending, no matter how complete the information content of each individual measurement $\hat{z}_{\sigma,t}$. The EKF accomplishes this through the Kalman gain K which is computed in (13). The Kalman gain, which is used to adjust the state and the error covariance in (15), is optimal in the sense that it minimizes the error covariance if certain conditions are met. Note that the inversion in (13) forms a ratio that reflects the relative uncertainties of the state and the measurement. Note too that the ratio is affected by the use of the measurement function Jacobian H. Because the Jacobian reflects the rate of change of each measurement with respect to the current state, it indicates a direction in state space along which a measurement could *possibly* affect the state. Because the gain is recomputed at each step with the appropriate measurement function and associated Jacobian, it inherently reflects the amount and direction of information provided by the individual constraint.

Welch 1997 at Section 3.1.3.

3.2 Autocalibration

The method we use for autocalibration involves *augmenting* the *main tracker filter* presented in section 3.1 to effectively implement a distinct *device filter*, a Kalman filter, for each source or sensor to be calibrated. (We use the word "device" here to include for example scene landmarks which can be thought of as passive sources, and cameras which are indeed sensors.) In general, any constant device-related parameters used by a measurement function $\hat{h}_\sigma(\bullet)$ from (8) are candidates for this autocalibration method. We assume that the parameters to be estimated are contained in the device parameter vectors $\hat{b}_t$ and $\hat{c}_t$, and we also present the case where both the source and sensor are to be calibrated since omission of one or the other is trivial. We note the following new convention.

$\widehat{\alpha}$ = augmented matrix/vector (wide hat)

Welch 1997 at Section 3.2

Exhibit E-7

CLAIM 1	Welch 1997

3.2.1 Device Filters

For each device (source, sensor, landmark, etc.) we create a distinct device filter as follows. Let $\hat{\pi}$ represent the corresponding device parameter vector and $n_\pi = \text{length}(\hat{\pi})$.

a. Allocate an $n_\pi \times 1$ state vector $\hat{x}_\pi$ for the device, initialize with the best *a priori* device parameter estimates, e.g. from design.

b. Allocate an $n_\pi \times n_\pi$ noise covariance matrix $Q_\pi(\delta t)$, initialize with the expected parameter variances.

c. Allocate an $n_\pi \times n_\pi$ error covariance matrix $P_\pi(t)$, initialize to indicate the level of confidence in the *a priori* device parameter estimates from (a) above.

Welch 1997 at Section 3.2.1.

3.2.2 Revised Tracking Algorithm

The algorithm for tracking with concurrent autocalibration is the same as that presented in section 3.1, with the following exceptions. After step (a) in the original algorithm, we form augmented versions of the state vector

$$\hat{x}(t-\delta t) = \left[\hat{x}^T(t-\delta t)\ \hat{x}_{b,t}^T(t-\delta t)\ \hat{x}_{c,t}^T(t-\delta t)\right]^T, \quad (18)$$

the error covariance matrix

$$\hat{P}(t-\delta t) = \begin{bmatrix} P(t-\delta t) & 0 & 0 \\ 0 & P_{b,t}(t-\delta t) & 0 \\ 0 & 0 & P_{c,t}(t-\delta t) \end{bmatrix}, \quad (19)$$

the state transition matrix

$$\hat{A}(\delta t) = \begin{bmatrix} A(\delta t) & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}, \quad (20)$$

and the process noise matrix

$$\hat{Q}(\delta t) = \begin{bmatrix} Q(\delta t) & 0 & 0 \\ 0 & Q_{b,t}(\delta t) & 0 \\ 0 & 0 & Q_{c,t}(\delta t) \end{bmatrix}. \quad (21)$$

Welch 1997 at Section 3.2.2.

Exhibit E-7

CLAIM 1	Welch 1997

We then follow steps (b)-(h) from the original algorithm, making the appropriate substitutions of (18)-(21), and noting that the measurement and Jacobian functions used in step (c) have become $\check{h}_\sigma(\widehat{x}(t))$ and $H_\sigma(\widehat{x}(t))$ because the estimates of parameters $\check{b}_t$ and $\check{\mathfrak{d}}_t$ ($\hat{x}_{\check{b},t}$ and $\hat{x}_{\check{c},t}$) are now contained in the augmented state vector $\widehat{x}$ per (18). After step (h) we finish by extracting and saving the device filter portions of the augmented state vector and error covariance matrix

$$\begin{aligned} \hat{x}_{\check{b},t}(t) &= \widehat{x}(t)[i..j] \\ P_{\check{b},t}(t) &= \widehat{P}(t)[i..j, i..j] \\ \hat{x}_{\check{c},t}(t) &= \widehat{x}(t)[k..l] \\ P_{\check{c},t}(t) &= \widehat{P}(t)[k..l, k..l] \end{aligned} \tag{22}$$

where

$$\begin{aligned} i &= n+1 \\ j &= n+n_{\check{b}} \\ k &= n+n_{\check{b}}+1 \\ l &= n+n_{\check{b}}+n_{\check{c}} \end{aligned}$$

and n, $n_{\check{b}}$, and $n_{\check{c}}$ are the dimensions of the state vectors for the main tracker filter, the source filter, and the sensor filter (respectively). We leave the main tracker filter state vector and error covariance matrix in their augmented counterparts, while we swap the device filter components in and out with each estimate. The result is that individual device filters are updated less frequently than the main tracker filter. The more a device is used, the more it is calibrated. If a device is never used, it is never calibrated.

Welch 1997 at Section 3.2.2.

Exhibit E-7

CLAIM 1	Welch 1997

With respect to added time complexity, the computations can again be optimized to eliminate operations on matrix and vector elements that are known to be zero: those places mentioned in section 3.1, and see (19)-(21). Also note that the size of and thus time for the matrix inversion in (13) has not changed. With respect to added space complexity, the autocalibration method requires storing a separate state vector and covariance matrix for each device—a fixed amount of (generally small) space per device. For example, consider autocalibrating the beacon (LED) positions for an optical tracking system with 3,000 beacons. For each beacon one would need 3 words for the beacon state (its 3D position), $3 \times 3 = 9$ words for the noise covariance matrix, and $3 \times 3 = 9$ words for the error covariance matrix. Assuming 8 bytes per word, this is only $3{,}000 \times 8 \times (3 + 9 + 9) = 504{,}000$ bytes.

Welch 1997 at Section 3.2.2.

3.2.3 Discussion

The ability to simultaneously estimate two dependent sets of unknowns (the target and device states) is made possible by several factors. First, the dynamics of the two sets are very different as would be reflected in the process noise matrices. We assume the target is undergoing some random (constant) acceleration, reflected in the noise parameter η of $Q(\delta t)$ in (6). Conversely, we assume the device parameters are constant, and so the elements of $Q_\pi(\delta t)$ for a source or sensor simply reflect any allowed variances in the corresponding parameters: usually zero or extremely small. In addition, while the target is expected to be moving, the filter expects the motion between any two estimations to closely correspond to the velocity estimates in the state (3). If the tracker estimate rate is high enough, poorly estimated device parameters will result in what appears to be almost instantaneous target motion. The increased rate of the SCAAT method allows such motion to be recognized as unlikely, and attributed to poorly estimated device parameters.

Welch 1997 at Section 3.2.3

3.3 Stability

Because the SCAAT method uses individual measurements with insufficient information, one might be concerned about the potential for instability or divergence. A linear system is said to be stable if its response to any input

CLAIM 1	Welch 1997
	tends to a finite steady value after the input is removed [24]. For the Kalman filter in general this is certainly not a new concern, and there are standard requirements and corresponding tests that ensure or detect stability (see [18], p. 132): a. The filter must be uniformly completely observable, b. the dynamic and measurement noise matrices in equations (6) and (9) must be bounded from above and below, and c. the dynamic behavior represented by in equation (2) must be bounded from above. As it turns out, these conditions and their standard tests are equally applicable to a SCAAT implementation. For the SCAAT method the conditions mean that the user dynamics between estimates must be bounded, the measurement noise must be bounded, one must incorporate a sufficient set of non-degenerate constraints over time. In particular, the constraints must be incorporated in less than 1/2 the time of the user motion time-constant in order to avoid tracking an alias of the true motion. In general these conditions are easily met for systems and circumstances that would otherwise be stable with a multiple-constraint implementation. A complete stability analysis is beyond the scope of this paper, and is presented in [47]. Welch 1997 at Section 3.3. [47] Greg Welch, 1996. "SCAAT: Incremental Tracking with Incomplete Information," University of North Carolina at Chapel Hill, doctoral dissertation, TR 96-051. Welch 1997 at References. *See also* Defendants' Invalidity Contentions for further discussion.
[1.a] an estimation subsystem; and	At least under Plaintiffs' apparent infringement theory, Welch 1997 discloses, either expressly or inherently, an estimation subsystem. In the alternative, this element would be obvious over Welch 1997 in light of the other references disclosed in Defendants' Invalidity Contentions and/or the knowledge of one of ordinary skill in the art. *See, e.g.*: **The Kalman filter [26] has been widely used for data fusion.** For example in navigation systems [17,30], virtual environment tracking systems [5,12,14], and in 3D scene modeling [20,42]. **However the SCAAT method represents a new approach to Kalman filter based multi-sensor data fusion.** Because constraints are intentionally incorporated one at a time, one can pick and choose which ones to add, and when to add them. This

CLAIM 1	Welch 1997
	means that information from different sensors or modalities can be woven together in a common, flexible, and expeditious fashion. Furthermore, one can use the approach to ensure that each estimate is computed from the most recently obtained constraint. Welch 1997 at Section 2.4. Consider for a moment the UNC hybrid landmark-magnetic presented at SIGGRAPH 96 [41]. This system uses an off-the-shelf Ascension magnetic tracking system along with a vision-based landmark recognition system to achieve superior synthetic and real image registration for augmented reality assisted medical procedures. The vision-based component attempts to identify and locate multiple known landmarks in a single image before applying a correction to the magnetic readings. A SCAAT implementation would instead identify and locate only one landmark per update, using a new image (frame) each time. Not only would this approach increase the frequency of landmark-based correction (given the necessary image processing) but it would offer the added benefit that unlike the implementation presented in [41], no special processing would be needed for the cases where the number of visible landmarks falls below the number *C* necessary to determine a complete position and orientation solution. The SCAAT implementation would simply cycle through any available landmarks, one at a time. Even with only one visible landmark the method would continue to operate as usual, using the information provided by the landmark sighting to refine the estimate where possible, while increasing the uncertainty where not. Welch 1997 at Section 2.4. **The SCAAT method employs a Kalman filter (KF) in an unusual fashion. The Kalman filter is a mathematical procedure that provides an efficient computational (recursive) method for the least-squares estimation of a linear system. It does so in a predictor-corrector fashion, predicting short-term (since the last estimate) changes in the state using a dynamic model, and then correcting them with a measurement and a corresponding measurement model. The extended Kalman filter (EKF) is a variation of the Kalman filter that supports estimation of nonlinear systems, e.g. 3D position and orientation tracking systems.** A basic introduction to the Kalman filter can be found in Chapter 1 of [31], while a more complete introductory discussion can be found in [40], which also contains some interesting historical narrative. More extensive references can be found in [7,18,24,28,31,46]. Welch 1997 at Section 3. The Kalman filter has been employed previously for virtual environment tracking estimation and prediction. For example see [2,5,12,14,42], and most recently [32]. In each of these cases however the filter was applied directly and only to the 6D pose estimates delivered by the off-the-shelf tracker. **The SCAAT approach could be applied**

Exhibit E-7

CLAIM 1	Welch 1997
	to either a hybrid system using off-the-shelf and/or custom trackers, or it could be employed by tracker developers to improve the existing systems for the end-user graphics community. Welch 1997 at Section 3. In this section we describe the method in a manner that does not imply a specific tracking system. (In section 3.4 we present experimental results of a specific implementation, a SCAAT wide-area optoelectronic tracking system.) In section 3.1 we describe the method for tracking, and in section 3.2 we describe one possible method for concurrent autocalibration. Welch 1997 at Section 3. Throughout we use the following conventions. x = scalar (lower case) $\vec{x}$ = general vector (lower case, arrow) indexed as $\vec{x}[r]$ $\hat{x}$ = filter estimate vector (lower case, hat) A = matrix (capital letters) indexed as $A[r, c]$ A^{-1} = matrix inverse I = the identity matrix β^{-} = matrix/vector *prediction* (super minus) β^{T} = matrix/vector transpose (super T) α_i = matrix/vector/scalar identifier (subscript) $E\{\bullet\}$ = mathematical expectation Welch 1997 at Section 3.

Exhibit E-7

CLAIM 1	Welch 1997
	3.1 Tracking **3.1.1 Main Tracker Filter** The use of a Kalman filter requires a mathematical (state-space) model for the dynamics of the process to be estimated, the target motion in this case. While several possible dynamic models and associated state configurations are possible, we have found a simple *position-velocity* model to suffice for the dynamics of our applications. In fact we use this same form of model, with different parameters, for all six of the position and orientation components $(x, y, z, \phi, \theta, \psi)$. Discussion of some other potential models and the associated trade-offs can be found in [7] pp. 415-420. Because our implementation is discrete with inter sample time δt we model the target's dynamic motion with the following linear difference equation: $$\dot{x}(t+\delta t) = A(\delta t)\dot{x}(t) + \dot{w}(\delta t). \quad (2)$$ Welch 1997 at Section 3.1.

Exhibit E-7

CLAIM 1	Welch 1997

In the standard model corresponding to equation (2), the n dimensional Kalman filter *state vector* $\hat{x}(t)$ would completely describe the target position and orientation at any time t. In practice we use a method similar to [2,6] and maintain the complete target orientation externally to the Kalman filter in order to avoid the nonlinearities associated with orientation computations. In the internal state vector $\hat{x}(t)$ we maintain the target position as the Cartesian coordinates (x, y, z), and the *incremental* orientation as small rotations (ϕ, θ, ψ) about the (x, y, z) axis. Externally we maintain the target orientation as the *external quaternion* $\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z))$. (See [9] for discussion of quaternions.) At each filter update step, the incremental orientations (ϕ, θ, ψ) are factored into the external quaternion $\hat{\alpha}$, and then zeroed as shown below. Thus the incremental orientations are linearized for the EKF, centered about zero. We maintain the derivatives of the target position and orientation internally, in the state vector $\hat{x}(t)$. We maintain the angular velocities internally because the angular velocities behave like orthogonal vectors and do not exhibit the nonlinearities of the angles themselves. The target state is then represented by the $n = 12$ element internal state vector

$$\hat{x} = \left[x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \phi\ \theta\ \psi\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}\right]^T \quad (3)$$

and the four-element external orientation quaternion

$$\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z)), \quad (4)$$

where the time designations have been omitted for clarity.

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 1	Welch 1997

The $n \times n$ *state transition matrix* $A(\delta t)$ in (2) projects the state forward from time t to time $t + \delta t$. For our linear model, the matrix implements the relationships

$$\begin{aligned} x(t+\delta t) &= x(t) + \dot{x}(t)\delta t \\ \dot{x}(t+\delta t) &= \dot{x}(t) \end{aligned} \tag{5}$$

and likewise for the remaining elements of (3).

The $n \times 1$ *process noise vector* $\vec{w}(\delta t)$ in (2) is a normally-distributed zero-mean sequence that represents the uncertainty in the target state over any time interval δt. The corresponding $n \times n$ *process noise covariance matrix* is given by

$$E\{\vec{w}(\delta t)\vec{w}^T(\delta t + \varepsilon)\} = \begin{cases} Q(\delta t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \tag{6}$$

Because our implementation is discrete with inter sample time δt, we can use the transfer function method illustrated by [7] pp. 221-222 to compute a *sampled* process noise covariance matrix. (Because the associated random processes are presumed to be time stationary, we present the process noise covariance matrix as a function of the inter-sample duration δt only.) The non-zero elements of $Q(\delta t)$ are given by

$$\begin{aligned} Q(\delta t)[i,i] &= \vec{\eta}[i]\frac{(\delta t)^3}{3} \\ Q(\delta t)[i,j] = Q(\delta t)[j,i] &= \vec{\eta}[i]\frac{(\delta t)^2}{2} \\ Q(\delta t)[j,j] &= \vec{\eta}[i](\delta t) \end{aligned} \tag{7}$$

for each pair

$$(i,j) \in \{(x,\dot{x}), (y,\dot{y}), (z,\dot{z}), (\phi,\dot{\phi}), (\theta,\dot{\theta}), (\psi,\dot{\psi})\}.$$

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 1	Welch 1997

The $\hat{\eta}[i]$ in (7) are the *correlation kernels* of the (assumed constant) noise sources presumed to be driving the dynamic model. We determined a set of values using Powell's method, and then used these in both simulation and our real implementation. The values can be "tuned" for different dynamics, though we have found that the tracker works well over a broad range of values.

The use of a Kalman filter requires not only a dynamic model as described above, but also a *measurement model* for each available type of measurement. The measurement model is used to predict the ideal noise-free response of each sensor and source pair, given the filter's current estimate of the target state as in equations (3) and (4).

It is the nature of the measurement models and indeed the actual sensor measurements that distinguishes a SCAAT Kalman filter from a well-constrained one.

For each sensor *type* σ we define the $m_\sigma \times 1$ *measurement vector* $\hat{z}_\sigma(t)$ and corresponding *measurement function* $\hat{h}_\sigma(\bullet)$ such that

$$\hat{z}_{\sigma,t} = \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t) + \hat{v}_\sigma(t) . \qquad (8)$$

Note that in the "purest" SCAAT implementation $m_\sigma = 1$ and the measurements are incorporated as single scalar values. However if it is not possible or necessary to isolate the measurements, e.g. to perform autocalibration, then multi-dimensional measurements can be incorporated also. Guidelines presented in [47] lead to the following heuristic for choosing the SCAAT Kalman filter measurement elements (constraints):

During each SCAAT Kalman filter measurement update one should observe a single sensor and source pair only.

For example, to incorporate magnetic tracker data as an end-user, $m_\sigma = 7$ for the three position and four orientation (quaternion)

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 1	Welch 1997

elements, while if the manufacturer were to use the SCAAT implementation, $m_\sigma = 3$ for each 3-axis electromagnetic response to a single excitation. For an image-based landmark tracker such as [41] the measurement function would, given estimates of the camera pose and a single landmark location, transform the landmark into camera space and then project it onto the camera image plane. In this case $m_\sigma = 2$ for the 2D image coordinates of the landmark.

The $m_\sigma \times 1$ *measurement noise vector* $\hat{\nu}_\sigma(t)$ in (8) is a normally-distributed zero-mean sequence that represents any random error (e.g. electrical noise) in the measurement. This parameter can be determined from component design specifications, and (or) confirmed by off-line measurement. For our simulations we did both. The corresponding $m_\sigma \times m_\sigma$ *measurement noise covariance matrix* is given by

$$E\{\hat{\nu}_\sigma(t)\hat{\nu}_\sigma^T(t+\varepsilon)\} = \begin{cases} R_\sigma(t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases} . \qquad (9)$$

For each measurement function $\hat{h}_\sigma(\bullet)$ we determine the corresponding Jacobian function

$$H_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t)[i,j] \equiv \frac{\partial}{\partial \hat{x}[j]} \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t)[i] , \qquad (10)$$

where $1 \le i \le m_\sigma$ and $1 \le j \le n$. Finally, we note the use of the standard (Kalman filter) $n \times n$ *error covariance matrix* $P(t)$ which maintains the covariance of the error in the estimated state.

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 1	Welch 1997

3.1.2 Tracking Algorithm

Given an initial state estimate $\hat{x}(0)$ and error covariance estimate $P(0)$, the SCAAT algorithm proceeds similarly to a conventional EKF, cycling through the following steps whenever a discrete measurement $\hat{z}_{\sigma,t}$ from some sensor (type σ) and source becomes available at time t:

a. Compute the time δt since the previous estimate.

b. Predict the state and error covariance.

$$\begin{aligned} \hat{x}^- &= A(\delta t)\hat{x}(t-\delta t) \\ P^- &= A(\delta t)P(t-\delta t)A^T(\delta t) + Q(\delta t) \end{aligned} \tag{11}$$

c. Predict the measurement and compute the corresponding Jacobian.

$$\begin{aligned} \hat{z} &= \hat{h}_\sigma(\hat{x}^-, \hat{b}_t, \vec{c}_t) \\ H &= H_\sigma(\hat{x}^-, \hat{b}_t, \hat{e}_t) \end{aligned} \tag{12}$$

d. Compute the *Kalman gain*.

$$K = P^- H^T (HP^- H^T + R_\sigma(t))^{-1} \tag{13}$$

e. Compute the *residual* between the actual sensor measurement $\hat{z}_{\sigma,t}$ and the predicted measurement from (12).

$$\vec{\Delta z} = \hat{z}_{\sigma,t} - \hat{z} \tag{14}$$

f. Correct the predicted tracker state estimate and error covariance from (11).

$$\begin{aligned} \hat{x}(t) &= \hat{x}^- + K\vec{\Delta z} \\ P(t) &= (I - KH)P^- \end{aligned} \tag{15}$$

Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 1	Welch 1997
	g. Update the external orientation of equation (4) per the change indicated by the (ϕ, θ, ψ) elements of the state.* $\Delta\hat{\alpha} = \text{quaternion}(\hat{x}[\phi], \hat{x}[\theta], \hat{x}[\psi])$ $\hat{\alpha} = \hat{\alpha} \otimes \Delta\hat{\alpha}$ (16) h. Zero the orientation elements of the state vector. $\hat{x}[\phi] = \hat{x}[\theta] = \hat{x}[\psi] = 0$ (17) The equations (11)-(17) may seem computationally complex, however they can be performed quite efficiently. The computations can be optimized to eliminate operations on matrix and vector elements that are known to be zero. For example, the elements of the Jacobian H in (12) that correspond to the velocities in the state $\hat{x}(t)$ will always be zero. In addition, the matrix inverted in the computation of K in (13) is of rank m_σ (2×2 for our example in section 3.4) which is smaller for a SCAAT filter than for a corresponding conventional EKF implementation. Finally, the increased data rate allows the use of the *small angle approximations* $\sin(\theta) = \theta$ and $\cos(\theta) = 1$ in $\check{h}_\sigma(\bullet)$ and $H_\sigma(\bullet)$. The total *per estimate* computation time can therefore actually be less than that of a corresponding conventional implementation. (We are able to execute the SCAAT filter computations, with the autocalibration computations discussed in the next section, in approximately 100µs on a 200 MHz PC-compatible computer.) Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 1	Welch 1997

3.1.3 Discussion

The key to the SCAAT method is the number of constraints provided by the measurement vector and measurement function in equation (8). For the 3D-tracking problem being solved, a unique solution requires $C = 6$ non-degenerate constraints to resolve six degrees of freedom. Because individual sensor measurements typically provide less than six constraints, conventional implementations usually construct a complete measurement vector

$$\hat{z}_t = \left[\hat{z}^T_{\sigma_1, t_1} \cdots \hat{z}^T_{\sigma_N, t_N} \right]^T$$

from some group of $N \geq C$ individual sensor measurements over time $t_1 \ldots t_N$, and *then* proceed to compute an estimate. Or a particular implementation may operate in a *moving-window* fashion, combining the most recent measurement with the $N-1$ previous measurements, possibly implementing a form of a finite-impulse-response filter. In any case, for such well-constrained systems complete observability is obtained at each step of the filter. Systems that collect measurements sequentially in this way inherently violate the simultaneity assumption, as well as increase the time δt between estimates.

Welch 1997 at Section 3.1.3.

In contrast, the SCAAT method blends individual measurements that each provide incomplete constraints into a complete state estimate. The EKF inherently provides the means for this blending, no matter how complete the information content of each individual measurement $\hat{z}_{\sigma, t}$. The EKF accomplishes this through the Kalman gain K which is computed in (13). The Kalman gain, which is used to adjust the state and the error covariance in (15), is optimal in the sense that it minimizes the error covariance if certain conditions are met. Note that the inversion in (13) forms a ratio that reflects the relative uncertainties of the state and the measurement. Note too that the ratio is affected by the use of the measurement function Jacobian H. Because the Jacobian reflects the rate of change of each measurement with respect to the current state, it indicates a direction in state space along which a measurement could *possibly* affect the state. Because the gain is recomputed at each step with the appropriate measurement function and associated Jacobian, it inherently reflects the amount and direction of information provided by the individual constraint.

Welch 1997 at Section 3.1.3.

Exhibit E-7

CLAIM 1	Welch 1997

3.2 Autocalibration

The method we use for autocalibration involves *augmenting* the *main tracker filter* presented in section 3.1 to effectively implement a distinct *device filter*, a Kalman filter, for each source or sensor to be calibrated. (We use the word "device" here to include for example scene landmarks which can be thought of as passive sources, and cameras which are indeed sensors.) In general, any constant device-related parameters used by a measurement function $\hat{h}_\sigma(\bullet)$ from (8) are candidates for this autocalibration method. We assume that the parameters to be estimated are contained in the device parameter vectors $\hat{b}_t$ and $\hat{c}_t$, and we also present the case where both the source and sensor are to be calibrated since omission of one or the other is trivial. We note the following new convention.

$\widehat{\alpha}$ = augmented matrix/vector (wide hat)

Welch 1997 at Section 3.2

3.2.1 Device Filters

For each device (source, sensor, landmark, etc.) we create a distinct device filter as follows. Let $\hat{\pi}$ represent the corresponding device parameter vector and $n_\pi = \text{length}(\hat{\pi})$.

a. Allocate an $n_\pi \times 1$ state vector $\hat{x}_\pi$ for the device, initialize with the best *a priori* device parameter estimates, e.g. from design.

b. Allocate an $n_\pi \times n_\pi$ noise covariance matrix $Q_\pi(\delta t)$, initialize with the expected parameter variances.

c. Allocate an $n_\pi \times n_\pi$ error covariance matrix $P_\pi(t)$, initialize to indicate the level of confidence in the *a priori* device parameter estimates from (a) above.

Welch 1997 at Section 3.2.1.

Exhibit E-7

CLAIM 1	Welch 1997

3.2.2 Revised Tracking Algorithm

The algorithm for tracking with concurrent autocalibration is the same as that presented in section 3.1, with the following exceptions. After step (a) in the original algorithm, we form augmented versions of the state vector

$$\hat{\mathbf{x}}(t-\delta t) = \left[\hat{\mathbf{x}}^T(t-\delta t)\ \hat{\mathbf{x}}_{b,t}^T(t-\delta t)\ \hat{\mathbf{x}}_{c,t}^T(t-\delta t)\right]^T, \quad (18)$$

the error covariance matrix

$$\hat{\mathbf{P}}(t-\delta t) = \begin{bmatrix} \mathbf{P}(t-\delta t) & 0 & 0 \\ 0 & \mathbf{P}_{b,t}(t-\delta t) & 0 \\ 0 & 0 & \mathbf{P}_{c,t}(t-\delta t) \end{bmatrix}, \quad (19)$$

the state transition matrix

$$\hat{\mathbf{A}}(\delta t) = \begin{bmatrix} \mathbf{A}(\delta t) & 0 & 0 \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I} \end{bmatrix}, \quad (20)$$

and the process noise matrix

$$\hat{\mathbf{Q}}(\delta t) = \begin{bmatrix} \mathbf{Q}(\delta t) & 0 & 0 \\ 0 & \mathbf{Q}_{b,t}(\delta t) & 0 \\ 0 & 0 & \mathbf{Q}_{c,t}(\delta t) \end{bmatrix}. \quad (21)$$

Welch 1997 at Section 3.2.2.

Exhibit E-7

CLAIM 1	Welch 1997

We then follow steps (b)-(h) from the original algorithm, making the appropriate substitutions of (18)-(21), and noting that the measurement and Jacobian functions used in step (c) have become $\check{h}_\sigma(\widehat{\vec{x}}(t))$ and $H_\sigma(\widehat{\vec{x}}(t))$ because the estimates of parameters $\check{b}_t$ and $\check{\vec{c}}_t$ ($\hat{x}_{\check{b},t}$ and $\hat{x}_{\check{c},t}$) are now contained in the augmented state vector $\widehat{\vec{x}}$ per (18). After step (h) we finish by extracting and saving the device filter portions of the augmented state vector and error covariance matrix

$$\begin{aligned} \hat{x}_{\check{b},t}(t) &= \widehat{\vec{x}}(t)[i..j] \\ P_{\check{b},t}(t) &= \widehat{P}(t)[i..j, i..j] \\ \hat{x}_{\check{c},t}(t) &= \widehat{\vec{x}}(t)[k..l] \\ P_{\check{c},t}(t) &= \widehat{P}(t)[k..l, k..l] \end{aligned} \tag{22}$$

where

$$\begin{aligned} i &= n+1 \\ j &= n+n_{\check{b}} \\ k &= n+n_{\check{b}}+1 \\ l &= n+n_{\check{b}}+n_{\check{c}} \end{aligned}$$

and n, $n_{\check{b}}$, and $n_{\check{c}}$ are the dimensions of the state vectors for the main tracker filter, the source filter, and the sensor filter (respectively). We leave the main tracker filter state vector and error covariance matrix in their augmented counterparts, while we swap the device filter components in and out with each estimate. The result is that individual device filters are updated less frequently than the main tracker filter. The more a device is used, the more it is calibrated. If a device is never used, it is never calibrated.

Welch 1997 at Section 3.2.2.

Exhibit E-7

CLAIM 1	Welch 1997

With respect to added time complexity, the computations can again be optimized to eliminate operations on matrix and vector elements that are known to be zero: those places mentioned in section 3.1, and see (19)-(21). Also note that the size of and thus time for the matrix inversion in (13) has not changed. With respect to added space complexity, the autocalibration method requires storing a separate state vector and covariance matrix for each device—a fixed amount of (generally small) space per device. For example, consider autocalibrating the beacon (LED) positions for an optical tracking system with 3,000 beacons. For each beacon one would need 3 words for the beacon state (its 3D position), $3 \times 3 = 9$ words for the noise covariance matrix, and $3 \times 3 = 9$ words for the error covariance matrix. Assuming 8 bytes per word, this is only $3{,}000 \times 8 \times (3 + 9 + 9) = 504{,}000$ bytes.

Welch 1997 at Section 3.2.2.

3.2.3 Discussion

The ability to simultaneously estimate two dependent sets of unknowns (the target and device states) is made possible by several factors. First, the dynamics of the two sets are very different as would be reflected in the process noise matrices. We assume the target is undergoing some random (constant) acceleration, reflected in the noise parameter η of $Q(\delta t)$ in (6). Conversely, we assume the device parameters are constant, and so the elements of $Q_\pi(\delta t)$ for a source or sensor simply reflect any allowed variances in the corresponding parameters: usually zero or extremely small. In addition, while the target is expected to be moving, the filter expects the motion between any two estimations to closely correspond to the velocity estimates in the state (3). If the tracker estimate rate is high enough, poorly estimated device parameters will result in what appears to be almost instantaneous target motion. The increased rate of the SCAAT method allows such motion to be recognized as unlikely, and attributed to poorly estimated device parameters.

Welch 1997 at Section 3.2.3

3.3 Stability

Because the SCAAT method uses individual measurements with insufficient information, one might be concerned about the potential for instability or divergence. A linear system is said to be stable if its response to any input tends

Exhibit E-7

CLAIM 1	Welch 1997
	to a finite steady value after the input is removed [24]. For the Kalman filter in general this is certainly not a new concern, and there are standard requirements and corresponding tests that ensure or detect stability (see [18], p. 132): a. The filter must be uniformly completely observable, b. the dynamic and measurement noise matrices in equations (6) and (9) must be bounded from above and below, and c. the dynamic behavior represented by in equation (2) must be bounded from above. As it turns out, these conditions and their standard tests are equally applicable to a SCAAT implementation. For the SCAAT method the conditions mean that the user dynamics between estimates must be bounded, the measurement noise must be bounded, one must incorporate a sufficient set of non-degenerate constraints over time. In particular, the constraints must be incorporated in less than 1/2 the time of the user motion time-constant in order to avoid tracking an alias of the true motion. In general these conditions are easily met for systems and circumstances that would otherwise be stable with a multiple-constraint implementation. A complete stability analysis is beyond the scope of this paper, and is presented in [47]. Welch 1997 at Section 3.3. [47] Greg Welch, 1996. "SCAAT: Incremental Tracking with Incomplete Information," University of North Carolina at Chapel Hill, doctoral dissertation, TR 96-051. Welch 1997 at References. *See* Disclosures with respect to Element 1.pre; *see also* Defendants' Invalidity Contentions for further discussion.
[1.b] a sensor subsystem coupled to the estimation subsystem and configured to provide configuration data to the estimation subsystem and to provide measurement information to the	At least under Plaintiffs' apparent infringement theory, Welch 1997 discloses, either expressly or inherently, a sensor subsystem coupled to the estimation subsystem and configured to provide configuration data to the estimation subsystem and to provide measurement information to the estimation subsystem for localizing an object. In the alternative, this element would be obvious over Welch 1997 in light of the other references disclosed in Defendants' Invalidity Contentions and/or the knowledge of one of ordinary skill in the art. *See, e.g.*:

Exhibit E-7

CLAIM 1	Welch 1997
estimation subsystem for localizing an object;	**1.1 Incomplete Information** The idea that one might build a tracking system that generates a new estimate with each individual sensor measurement or *observation* is a very interesting one. After all, individual observations usually provide only partial information about a user's complete state (pose), i.e. they are "incomplete" observations. For example, for a camera observing landmarks in a scene, only limited information is obtained from observations of any single landmark. In terms of control theory, a system designed to operate with only such incomplete measurements is characterized as *unobservable* because the user state cannot be observed (determined) from the measurements. The notion of observability can also be described in terms of constraints on the unknown parameters of the system being estimated, e.g. constraints on the unknown elements of the system state. Given a particular system, and the corresponding set of unknowns that are to be estimated, let C be defined as the minimal number of independent simultaneous constraints necessary to uniquely determine a solution, let N be the number actually used to generate a new estimate, and let N_{ind} be the number of *independent* constraints that can be formed from the N constraints. For any $N \geq N_{ind}$ constraints, if $N_{ind} = C$ the problem is *well constrained*, if $N_{ind} > C$ it is *over constrained*, and if $N_{ind} < C$ it is *under-constrained*. (See Figure 1.) Welch 1997 at Section 1.1.

Exhibit E-7

CLAIM 1	Welch 1997

1.2 Landmark Tracking

Consider for example a system in which a single camera is used to observe known scene points to determine the camera position and orientation. In this case, the constraints provided by the observations are multi-dimensional: 2D image coordinates of 3D scene points. Given the internal camera parameters, a set of four known coplanar scene points, and the corresponding image coordinates, the camera position and orientation can be uniquely determined in closed-form [16]. In other words if $N = C = 4$ constraints (2D image points) are used to estimate the camera position and orientation, the system is completely observable. On the other hand, if $N < C$ then there are multiple solutions. For example with only $N = 3$ non-collinear points, there are up to 4 solutions. Even worse, with $N = 2$ or $N = 1$ points, there are infinite combinations of position and orientation that could result in the same camera images.

Welch 1997 at Section 1.2.

In general, for closed-form tracking approaches, a well or over-constrained system with $N \geq C$ is observable, an under-constrained system with $N < C$ is not. Therefore, if the individual observations provide only partial information, i.e. the measurements provide insufficient constraints, then multiple devices or landmarks must be excited and (or) sensed prior to estimating a solution. Sometimes the necessary observations can be obtained simultaneously, and sometimes they can not. Magnetic trackers such as those made by Polhemus and Ascension perform three *sequential* source excitations, each in conjunction with a complete sensor unit observation. And while a camera can indeed observe multiple landmarks simultaneously in a single image, the image processing to identify and locate the individual landmarks must be done sequentially for a single CPU system. If the landmarks can move independently over time, for example if they are artificial marks placed on the skin of an ultrasound patient for the purpose of landmark-based tracking [41], batch processing of the landmarks can reduce the effectiveness of the system. A SCAAT implementation might grab an image, extract a *single* landmark, update the estimates of both the camera *and* landmark positions, and then throw-away the image. In this way estimates are generated faster and with the most recent landmark configurations.

Welch 1997 at Section 1.2.

Exhibit E-7

CLAIM 1	Welch 1997

1.3 Putting the Pieces Together

Given a tracker that uses multiple constraints that are each individually incomplete, a *measurement model* for any one of incomplete constraints would be characterized as *locally unobservable*. Such a system must incorporate a sufficient set of these incomplete constraints so that the resulting overall system is observable. The corresponding aggregate measurement model can then be characterized as *globally observable*. Global observability can be obtained over *space* or over *time*. The SCAAT method adopts the latter scheme, even in some cases where the former is possible.

Welch 1997 at Section 1.3.

The Kalman filter [26] has been widely used for data fusion. For example in navigation systems [17,30], virtual environment tracking systems [5,12,14], and in 3D scene modeling [20,42]. However the SCAAT method represents a new approach to Kalman filter based multi-sensor data fusion. Because constraints are intentionally incorporated one at a time, one can pick and choose which ones to add, and when to add them. This means that information from different sensors or modalities can be woven together in a common, flexible, and expeditious fashion. Furthermore, one can use the approach to ensure that each estimate is computed from the most recently obtained constraint.
Welch 1997 at Section 2.4.

Consider for a moment the UNC hybrid landmark-magnetic presented at SIGGRAPH 96 [41]. This system uses an off-the-shelf Ascension magnetic tracking system along with a vision-based landmark recognition system to achieve superior synthetic and real image registration for augmented reality assisted medical procedures. The vision-based component attempts to identify and locate multiple known landmarks in a single image before applying a correction to the magnetic readings. **A SCAAT implementation would instead identify and locate only one landmark per update, using a new image (frame) each time. Not only would this approach increase the frequency of landmark-based correction (given the necessary image processing) but it would offer the added benefit that unlike the implementation presented in [41], no special processing would be needed for the cases where the number of visible landmarks falls below the number *C* necessary to determine a complete position and orientation solution. The SCAAT implementation would simply cycle through any available landmarks, one at a time. Even with only one visible landmark the method would continue to operate as usual, using the information provided by the landmark sighting to refine the estimate where**

Exhibit E-7

CLAIM 1	Welch 1997
	possible, while increasing the uncertainty where not. Welch 1997 at Section 2.4. The SCAAT method employs a Kalman filter (KF) in an unusual fashion. The Kalman filter is a mathematical procedure that provides an efficient computational (recursive) method for the least-squares estimation of a linear system. It does so in a predictor-corrector fashion, predicting short-term (since the last estimate) changes in the state using a dynamic model, and then correcting them with a measurement and a corresponding measurement model. The extended Kalman filter (EKF) is a variation of the Kalman filter that supports estimation of nonlinear systems, e.g. 3D position and orientation tracking systems. A basic introduction to the Kalman filter can be found in Chapter 1 of [31], while a more complete introductory discussion can be found in [40], which also contains some interesting historical narrative. More extensive references can be found in [7,18,24,28,31,46]. Welch 1997 at Section 3. The Kalman filter has been employed previously for virtual environment tracking estimation and prediction. For example see [2,5,12,14,42], and most recently [32]. In each of these cases however the filter was applied directly and only to the 6D pose estimates delivered by the off-the-shelf tracker. The SCAAT approach could be applied to either a hybrid system using off-the-shelf and/or custom trackers, or it could be employed by tracker developers to improve the existing systems for the end-user graphics community. Welch 1997 at Section 3. In this section we describe the method in a manner that does not imply a specific tracking system. (In section 3.4 we present experimental results of a specific implementation, a SCAAT wide-area optoelectronic tracking system.) In section 3.1 we describe the method for tracking, and in section 3.2 we describe one possible method for concurrent autocalibration. Welch 1997 at Section 3.

Exhibit E-7

CLAIM 1	Welch 1997

Throughout we use the following conventions.

x = scalar (lower case)

$\vec{x}$ = general vector (lower case, arrow) indexed as $\vec{x}[r]$

$\hat{x}$ = filter estimate vector (lower case, hat)

A = matrix (capital letters) indexed as $A[r, c]$

A^{-1} = matrix inverse

I = the identity matrix

β^{-} = matrix/vector *prediction* (super minus)

β^{T} = matrix/vector transpose (super T)

α_i = matrix/vector/scalar identifier (subscript)

$E\{\bullet\}$ = mathematical expectation

Welch 1997 at Section 3.

3.1 Tracking

3.1.1 Main Tracker Filter

The use of a Kalman filter requires a mathematical (state-space) model for the dynamics of the process to be estimated, the target motion in this case. While several possible dynamic models and associated state configurations are possible, we have found a simple *position-velocity* model to suffice for the dynamics of our applications. In fact we use this same form of model, with different parameters, for all six of the position and orientation components $(x, y, z, \phi, \theta, \psi)$. Discussion of some other potential models and the associated trade-offs can be found in [7] pp. 415-420. Because our implementation is discrete with inter sample time δt we model the target's dynamic motion with the following linear difference equation:

$$\vec{x}(t+\delta t) = A(\delta t)\vec{x}(t) + \vec{w}(\delta t). \qquad (2)$$

Welch 1997 at Section 3.1.

Exhibit E-7

CLAIM 1	Welch 1997

In the standard model corresponding to equation (2), the n dimensional Kalman filter *state vector* $\hat{x}(t)$ would completely describe the target position and orientation at any time t. In practice we use a method similar to [2,6] and maintain the complete target orientation externally to the Kalman filter in order to avoid the nonlinearities associated with orientation computations. In the internal state vector $\hat{x}(t)$ we maintain the target position as the Cartesian coordinates (x, y, z), and the *incremental* orientation as small rotations (ϕ, θ, ψ) about the (x, y, z) axis. Externally we maintain the target orientation as the *external quaternion* $\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z))$. (See [9] for discussion of quaternions.) At each filter update step, the incremental orientations (ϕ, θ, ψ) are factored into the external quaternion $\hat{\alpha}$, and then zeroed as shown below. Thus the incremental orientations are linearized for the EKF, centered about zero. We maintain the derivatives of the target position and orientation internally, in the state vector $\hat{x}(t)$. We maintain the angular velocities internally because the angular velocities behave like orthogonal vectors and do not exhibit the nonlinearities of the angles themselves. The target state is then represented by the $n = 12$ element internal state vector

$$\hat{x} = \left[x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \phi\ \theta\ \psi\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}\right]^T \quad (3)$$

and the four-element external orientation quaternion

$$\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z)), \quad (4)$$

where the time designations have been omitted for clarity.

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 1	Welch 1997

The $n \times n$ *state transition matrix* $A(\delta t)$ in (2) projects the state forward from time t to time $t + \delta t$. For our linear model, the matrix implements the relationships

$$\begin{aligned} x(t+\delta t) &= x(t) + \dot{x}(t)\delta t \\ \dot{x}(t+\delta t) &= \dot{x}(t) \end{aligned} \qquad (5)$$

and likewise for the remaining elements of (3).

The $n \times 1$ *process noise vector* $\vec{w}(\delta t)$ in (2) is a normally-distributed zero-mean sequence that represents the uncertainty in the target state over any time interval δt. The corresponding $n \times n$ *process noise covariance matrix* is given by

$$E\{\vec{w}(\delta t)\vec{w}^T(\delta t + \varepsilon)\} = \begin{cases} Q(\delta t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \qquad (6)$$

Because our implementation is discrete with inter sample time δt, we can use the transfer function method illustrated by [7] pp. 221-222 to compute a *sampled* process noise covariance matrix. (Because the associated random processes are presumed to be time stationary, we present the process noise covariance matrix as a function of the inter-sample duration δt only.) The non-zero elements of $Q(\delta t)$ are given by

$$\begin{aligned} Q(\delta t)[i,i] &= \vec{\eta}[i]\frac{(\delta t)^3}{3} \\ Q(\delta t)[i,j] = Q(\delta t)[j,i] &= \vec{\eta}[i]\frac{(\delta t)^2}{2} \\ Q(\delta t)[j,j] &= \vec{\eta}[i](\delta t) \end{aligned} \qquad (7)$$

for each pair

$$(i,j) \in \{(x,\dot{x}), (y,\dot{y}), (z,\dot{z}), (\phi,\dot{\phi}), (\theta,\dot{\theta}), (\psi,\dot{\psi})\}.$$

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 1	Welch 1997

The $\hat{\eta}[i]$ in (7) are the *correlation kernels* of the (assumed constant) noise sources presumed to be driving the dynamic model. We determined a set of values using Powell's method, and then used these in both simulation and our real implementation. The values can be "tuned" for different dynamics, though we have found that the tracker works well over a broad range of values.

The use of a Kalman filter requires not only a dynamic model as described above, but also a *measurement model* for each available type of measurement. The measurement model is used to predict the ideal noise-free response of each sensor and source pair, given the filter's current estimate of the target state as in equations (3) and (4).

It is the nature of the measurement models and indeed the actual sensor measurements that distinguishes a SCAAT Kalman filter from a well-constrained one.

For each sensor *type* σ we define the $m_\sigma \times 1$ *measurement vector* $\hat{z}_\sigma(t)$ and corresponding *measurement function* $\hat{h}_\sigma(\bullet)$ such that

$$\hat{z}_{\sigma,t} = \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t) + \hat{v}_\sigma(t) . \qquad (8)$$

Note that in the "purest" SCAAT implementation $m_\sigma = 1$ and the measurements are incorporated as single scalar values. However if it is not possible or necessary to isolate the measurements, e.g. to perform autocalibration, then multi-dimensional measurements can be incorporated also. Guidelines presented in [47] lead to the following heuristic for choosing the SCAAT Kalman filter measurement elements (constraints):

During each SCAAT Kalman filter measurement update one should observe a single sensor and source pair only.

For example, to incorporate magnetic tracker data as an end-user, $m_\sigma = 7$ for the three position and four orientation (quaternion)

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 1	Welch 1997

elements, while if the manufacturer were to use the SCAAT implementation, $m_\sigma = 3$ for each 3-axis electromagnetic response to a single excitation. For an image-based landmark tracker such as [41] the measurement function would, given estimates of the camera pose and a single landmark location, transform the landmark into camera space and then project it onto the camera image plane. In this case $m_\sigma = 2$ for the 2D image coordinates of the landmark.

The $m_\sigma \times 1$ *measurement noise vector* $\hat{\nu}_\sigma(t)$ in (8) is a normally-distributed zero-mean sequence that represents any random error (e.g. electrical noise) in the measurement. This parameter can be determined from component design specifications, and (or) confirmed by off-line measurement. For our simulations we did both. The corresponding $m_\sigma \times m_\sigma$ *measurement noise covariance matrix* is given by

$$E\{\hat{\nu}_\sigma(t)\hat{\nu}_\sigma^T(t+\varepsilon)\} = \begin{cases} R_\sigma(t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases} . \qquad (9)$$

For each measurement function $\hat{h}_\sigma(\bullet)$ we determine the corresponding Jacobian function

$$H_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t)[i,j] \equiv \frac{\partial}{\partial \hat{x}[j]} \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t)[i] , \qquad (10)$$

where $1 \le i \le m_\sigma$ and $1 \le j \le n$. Finally, we note the use of the standard (Kalman filter) $n \times n$ *error covariance matrix* $P(t)$ which maintains the covariance of the error in the estimated state.

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 1	Welch 1997

3.1.2 Tracking Algorithm

Given an initial state estimate $\hat{x}(0)$ and error covariance estimate $P(0)$, the SCAAT algorithm proceeds similarly to a conventional EKF, cycling through the following steps whenever a discrete measurement $\vec{z}_{\sigma,t}$ from some sensor (type σ) and source becomes available at time t:

a. Compute the time δt since the previous estimate.

b. Predict the state and error covariance.

$$\begin{aligned} \hat{x}^- &= A(\delta t)\hat{x}(t-\delta t) \\ P^- &= A(\delta t)P(t-\delta t)A^T(\delta t) + Q(\delta t) \end{aligned} \tag{11}$$

c. Predict the measurement and compute the corresponding Jacobian.

$$\begin{aligned} \hat{z} &= \vec{h}_\sigma(\hat{x}^-, \vec{b}_t, \vec{c}_t) \\ H &= H_\sigma(\hat{x}^-, \vec{b}_t, \vec{e}_t) \end{aligned} \tag{12}$$

d. Compute the *Kalman gain*.

$$K = P^- H^T (HP^- H^T + R_\sigma(t))^{-1} \tag{13}$$

e. Compute the *residual* between the actual sensor measurement $\vec{z}_{\sigma,t}$ and the predicted measurement from (12).

$$\vec{\Delta z} = \vec{z}_{\sigma,t} - \hat{z} \tag{14}$$

f. Correct the predicted tracker state estimate and error covariance from (11).

$$\begin{aligned} \hat{x}(t) &= \hat{x}^- + K\vec{\Delta z} \\ P(t) &= (I - KH)P^- \end{aligned} \tag{15}$$

Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 1	Welch 1997
	g. Update the external orientation of equation (4) per the change indicated by the (ϕ, θ, ψ) elements of the state.* $\Delta\hat{\alpha} = \text{quaternion}(\hat{x}[\phi], \hat{x}[\theta], \hat{x}[\psi])$ $\hat{\alpha} = \hat{\alpha} \otimes \Delta\hat{\alpha}$ (16) h. Zero the orientation elements of the state vector. $\hat{x}[\phi] = \hat{x}[\theta] = \hat{x}[\psi] = 0$ (17) The equations (11)-(17) may seem computationally complex, however they can be performed quite efficiently. The computations can be optimized to eliminate operations on matrix and vector elements that are known to be zero. For example, the elements of the Jacobian H in (12) that correspond to the velocities in the state $\hat{x}(t)$ will always be zero. In addition, the matrix inverted in the computation of K in (13) is of rank m_σ (2×2 for our example in section 3.4) which is smaller for a SCAAT filter than for a corresponding conventional EKF implementation. Finally, the increased data rate allows the use of the *small angle approximations* $\sin(\theta) = \theta$ and $\cos(\theta) = 1$ in $\check{h}_\sigma(\bullet)$ and $H_\sigma(\bullet)$. The total *per estimate* computation time can therefore actually be less than that of a corresponding conventional implementation. (We are able to execute the SCAAT filter computations, with the autocalibration computations discussed in the next section, in approximately 100µs on a 200 MHz PC-compatible computer.) Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 1	Welch 1997

3.1.3 Discussion

The key to the SCAAT method is the number of constraints provided by the measurement vector and measurement function in equation (8). For the 3D-tracking problem being solved, a unique solution requires $C = 6$ non-degenerate constraints to resolve six degrees of freedom. Because individual sensor measurements typically provide less than six constraints, conventional implementations usually construct a complete measurement vector

$$\hat{z}_t = \left[\hat{z}^T_{\sigma_1, t_1} \cdots \hat{z}^T_{\sigma_N, t_N} \right]^T$$

from some group of $N \geq C$ individual sensor measurements over time $t_1 \ldots t_N$, and *then* proceed to compute an estimate. Or a particular implementation may operate in a *moving-window* fashion, combining the most recent measurement with the $N-1$ previous measurements, possibly implementing a form of a finite-impulse-response filter. In any case, for such well-constrained systems complete observability is obtained at each step of the filter. Systems that collect measurements sequentially in this way inherently violate the simultaneity assumption, as well as increase the time δt between estimates.

Welch 1997 at Section 3.1.3.

In contrast, the SCAAT method blends individual measurements that each provide incomplete constraints into a complete state estimate. The EKF inherently provides the means for this blending, no matter how complete the information content of each individual measurement $\hat{z}_{\sigma, t}$. The EKF accomplishes this through the Kalman gain K which is computed in (13). The Kalman gain, which is used to adjust the state and the error covariance in (15), is optimal in the sense that it minimizes the error covariance if certain conditions are met. Note that the inversion in (13) forms a ratio that reflects the relative uncertainties of the state and the measurement. Note too that the ratio is affected by the use of the measurement function Jacobian H. Because the Jacobian reflects the rate of change of each measurement with respect to the current state, it indicates a direction in state space along which a measurement could *possibly* affect the state. Because the gain is recomputed at each step with the appropriate measurement function and associated Jacobian, it inherently reflects the amount and direction of information provided by the individual constraint.

Welch 1997 at Section 3.1.3.

CLAIM 1	Welch 1997
	3.3 Stability Because the SCAAT method uses individual measurements with insufficient information, one might be concerned about the potential for instability or divergence. A linear system is said to be stable if its response to any input tends to a finite steady value after the input is removed [24]. For the Kalman filter in general this is certainly not a new concern, and there are standard requirements and corresponding tests that ensure or detect stability (see [18], p. 132): a. The filter must be uniformly completely observable, b. the dynamic and measurement noise matrices in equations (6) and (9) must be bounded from above and below, and c. the dynamic behavior represented by in equation (2) must be bounded from above. As it turns out, these conditions and their standard tests are equally applicable to a SCAAT implementation. For the SCAAT method the conditions mean that the user dynamics between estimates must be bounded, the measurement noise must be bounded, one must incorporate a sufficient set of non-degenerate constraints over time. In particular, the constraints must be incorporated in less than 1/2 the time of the user motion time-constant in order to avoid tracking an alias of the true motion. In general these conditions are easily met for systems and circumstances that would otherwise be stable with a multiple-constraint implementation. A complete stability analysis is beyond the scope of this paper, and is presented in [47]. Welch 1997 at Section 3.3. [47] Greg Welch, 1996. "SCAAT: Incremental Tracking with Incomplete Information," University of North Carolina at Chapel Hill, doctoral dissertation, TR 96-051. Welch 1997 at References. *See* Disclosures with respect to Element 1.a; *see also* Defendants' Invalidity Contentions for further discussion.
[1.c] wherein the estimation subsystem is configured to update a location estimate for the object based on	At least under Plaintiffs' apparent infringement theory, Welch 1997 discloses, either expressly or inherently, wherein the estimation subsystem is configured to update a location estimate for the object based on configuration data and measurement information accepted from the sensor subsystem. In the alternative, this element would be obvious over Welch 1997 in light of the other references disclosed in Defendants' Invalidity Contentions and/or the knowledge of one of ordinary skill in the art.

Exhibit E-7

CLAIM 1	Welch 1997
configuration data and measurement information accepted from the sensor subsystem.	*See, e.g.*: **We present a promising new mathematical method for tracking a user's pose (position and orientation) for interactive computer graphics. The method, which is applicable to a wide variety of both commercial and experimental systems, improves accuracy by properly assimilating sequential observations, filtering sensor measurements, and by concurrently autocalibrating source and sensor devices. It facilitates user motion prediction, multisensor data fusion, and higher report rates with lower latency than previous methods.** **Tracking systems determine the user's pose by measuring signals from low-level hardware sensors.** For reasons of physics and economics, most systems make multiple sequential measurements which are then combined to produce a single tracker report. For example, commercial magnetic trackers using the SPASYN (Space Synchro) system sequentially measure three magnetic vectors and then combine them mathematically to produce a report of the sensor pose. **Our new approach produces tracker reports as each new low-level sensor measurement is made rather than waiting to form a complete collection of observations.** Because single observations under-constrain the mathematical solution, we refer to our approach as single-constraint-at-a-time or SCAAT tracking. **The key is that the single observations provide some information about the user's state, and thus can be used to incrementally improve a previous estimate. We recursively apply this principle, incorporating new sensor data as soon as it is measured. With this approach we are able to generate estimates more frequently, with less latency, and with improved accuracy. We present results from both an actual implementation, and from extensive simulations.** Welch 1997 at Abstract. The Kalman filter [26] has been widely used for data fusion. For example in navigation systems [17,30], virtual environment tracking systems [5,12,14], and in 3D scene modeling [20,42]. However the SCAAT method represents a new approach to Kalman filter based multi-sensor data fusion. Because constraints are intentionally incorporated one at a time, one can pick and choose which ones to add, and when to add them. This means that information from different sensors or modalities can be woven together in a common, flexible, and expeditious fashion. Furthermore, one can use the approach to ensure that each estimate is computed from the most recently obtained constraint. Welch 1997 at Section 2.4.

Exhibit E-7

CLAIM 1	Welch 1997
	Consider for a moment the UNC hybrid landmark-magnetic presented at SIGGRAPH 96 [41]. This system uses an off-the-shelf Ascension magnetic tracking system along with a vision-based landmark recognition system to achieve superior synthetic and real image registration for augmented reality assisted medical procedures. The vision-based component attempts to identify and locate multiple known landmarks in a single image before applying a correction to the magnetic readings. **A SCAAT implementation would instead identify and locate only one landmark per update, using a new image (frame) each time. Not only would this approach increase the frequency of landmark-based correction (given the necessary image processing) but it would offer the added benefit that unlike the implementation presented in [41], no special processing would be needed for the cases where the number of visible landmarks falls below the number *C* necessary to determine a complete position and orientation solution. The SCAAT implementation would simply cycle through any available landmarks, one at a time. Even with only one visible landmark the method would continue to operate as usual, using the information provided by the landmark sighting to refine the estimate where possible, while increasing the uncertainty where not.** Welch 1997 at Section 2.4. The SCAAT method employs a Kalman filter (KF) in an unusual fashion. The Kalman filter is a mathematical procedure that provides an efficient computational (recursive) method for the least-squares estimation of a linear system. **It does so in a predictor-corrector fashion, predicting short-term (since the last estimate) changes in the state using a dynamic model, and then correcting them with a measurement and a corresponding measurement model. The extended Kalman filter (EKF) is a variation of the Kalman filter that supports estimation of nonlinear systems, e.g. 3D position and orientation tracking systems.** A basic introduction to the Kalman filter can be found in Chapter 1 of [31], while a more complete introductory discussion can be found in [40], which also contains some interesting historical narrative. More extensive references can be found in [7,18,24,28,31,46]. Welch 1997 at Section 3. The Kalman filter has been employed previously for virtual environment tracking estimation and prediction. For example see [2,5,12,14,42], and most recently [32]. In each of these cases however the filter was applied directly and only to the 6D pose estimates delivered by the off-the-shelf tracker. The SCAAT approach could be applied to either a hybrid system using off-the-shelf and/or custom trackers, or it could be employed by tracker developers to improve the existing systems for the end-user graphics community. Welch 1997 at Section 3.

Exhibit E-7

CLAIM 1	Welch 1997
	In this section we describe the method in a manner that does not imply a specific tracking system. (In section 3.4 we present experimental results of a specific implementation, a SCAAT wide-area optoelectronic tracking system.) In section 3.1 we describe the method for tracking, and in section 3.2 we describe one possible method for concurrent autocalibration. Welch 1997 at Section 3. Throughout we use the following conventions. x = scalar (lower case) $\vec{x}$ = general vector (lower case, arrow) indexed as $\vec{x}[r]$ $\hat{x}$ = filter estimate vector (lower case, hat) A = matrix (capital letters) indexed as $A[r, c]$ A^{-1} = matrix inverse I = the identity matrix β^{-} = matrix/vector *prediction* (super minus) β^{T} = matrix/vector transpose (super T) α_i = matrix/vector/scalar identifier (subscript) $E\{\bullet\}$ = mathematical expectation Welch 1997 at Section 3.

Exhibit E-7

CLAIM 1	Welch 1997
	3.1 Tracking **3.1.1 Main Tracker Filter** The use of a Kalman filter requires a mathematical (state-space) model for the dynamics of the process to be estimated, the target motion in this case. While several possible dynamic models and associated state configurations are possible, we have found a simple *position-velocity* model to suffice for the dynamics of our applications. In fact we use this same form of model, with different parameters, for all six of the position and orientation components $(x, y, z, \phi, \theta, \psi)$. Discussion of some other potential models and the associated trade-offs can be found in [7] pp. 415-420. Because our implementation is discrete with inter sample time δt we model the target's dynamic motion with the following linear difference equation: $$\hat{x}(t+\delta t) = A(\delta t)\hat{x}(t) + \hat{w}(\delta t). \quad (2)$$ Welch 1997 at Section 3.1.

Exhibit E-7

CLAIM 1	Welch 1997

In the standard model corresponding to equation (2), the n dimensional Kalman filter *state vector* $\hat{x}(t)$ would completely describe the target position and orientation at any time t. In practice we use a method similar to [2,6] and maintain the complete target orientation externally to the Kalman filter in order to avoid the nonlinearities associated with orientation computations. In the internal state vector $\hat{x}(t)$ we maintain the target position as the Cartesian coordinates (x, y, z), and the *incremental* orientation as small rotations (ϕ, θ, ψ) about the (x, y, z) axis. Externally we maintain the target orientation as the *external quaternion* $\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z))$. (See [9] for discussion of quaternions.) At each filter update step, the incremental orientations (ϕ, θ, ψ) are factored into the external quaternion $\hat{\alpha}$, and then zeroed as shown below. Thus the incremental orientations are linearized for the EKF, centered about zero. We maintain the derivatives of the target position and orientation internally, in the state vector $\hat{x}(t)$. We maintain the angular velocities internally because the angular velocities behave like orthogonal vectors and do not exhibit the nonlinearities of the angles themselves. The target state is then represented by the $n = 12$ element internal state vector

$$\hat{x} = \left[x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \phi\ \theta\ \psi\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}\right]^T \quad (3)$$

and the four-element external orientation quaternion

$$\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z)), \quad (4)$$

where the time designations have been omitted for clarity.

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 1	Welch 1997

The $n \times n$ *state transition matrix* $A(\delta t)$ in (2) projects the state forward from time t to time $t + \delta t$. For our linear model, the matrix implements the relationships

$$\begin{aligned} x(t+\delta t) &= x(t) + \dot{x}(t)\delta t \\ \dot{x}(t+\delta t) &= \dot{x}(t) \end{aligned} \qquad (5)$$

and likewise for the remaining elements of (3).

The $n \times 1$ *process noise vector* $\vec{w}(\delta t)$ in (2) is a normally-distributed zero-mean sequence that represents the uncertainty in the target state over any time interval δt. The corresponding $n \times n$ *process noise covariance matrix* is given by

$$E\{\vec{w}(\delta t)\vec{w}^T(\delta t + \varepsilon)\} = \begin{cases} Q(\delta t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \qquad (6)$$

Because our implementation is discrete with inter sample time δt, we can use the transfer function method illustrated by [7] pp. 221-222 to compute a *sampled* process noise covariance matrix. (Because the associated random processes are presumed to be time stationary, we present the process noise covariance matrix as a function of the inter-sample duration δt only.) The non-zero elements of $Q(\delta t)$ are given by

$$\begin{aligned} Q(\delta t)[i,i] &= \vec{\eta}[i]\frac{(\delta t)^3}{3} \\ Q(\delta t)[i,j] = Q(\delta t)[j,i] &= \vec{\eta}[i]\frac{(\delta t)^2}{2} \\ Q(\delta t)[j,j] &= \vec{\eta}[i](\delta t) \end{aligned} \qquad (7)$$

for each pair

$$(i,j) \in \{(x,\dot{x}), (y,\dot{y}), (z,\dot{z}), (\phi,\dot{\phi}), (\theta,\dot{\theta}), (\psi,\dot{\psi})\}.$$

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 1	Welch 1997

The $\hat{\eta}[i]$ in (7) are the *correlation kernels* of the (assumed constant) noise sources presumed to be driving the dynamic model. We determined a set of values using Powell's method, and then used these in both simulation and our real implementation. The values can be "tuned" for different dynamics, though we have found that the tracker works well over a broad range of values.

The use of a Kalman filter requires not only a dynamic model as described above, but also a *measurement model* for each available type of measurement. The measurement model is used to predict the ideal noise-free response of each sensor and source pair, given the filter's current estimate of the target state as in equations (3) and (4).

It is the nature of the measurement models and indeed the actual sensor measurements that distinguishes a SCAAT Kalman filter from a well-constrained one.

For each sensor *type* σ we define the $m_\sigma \times 1$ *measurement vector* $\hat{z}_\sigma(t)$ and corresponding *measurement function* $\hat{h}_\sigma(\bullet)$ such that

$$\hat{z}_{\sigma,t} = \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t) + \hat{v}_\sigma(t) . \qquad (8)$$

Note that in the "purest" SCAAT implementation $m_\sigma = 1$ and the measurements are incorporated as single scalar values. However if it is not possible or necessary to isolate the measurements, e.g. to perform autocalibration, then multi-dimensional measurements can be incorporated also. Guidelines presented in [47] lead to the following heuristic for choosing the SCAAT Kalman filter measurement elements (constraints):

During each SCAAT Kalman filter measurement update one should observe a single sensor and source pair only.

For example, to incorporate magnetic tracker data as an end-user, $m_\sigma = 7$ for the three position and four orientation (quaternion)

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 1	Welch 1997

elements, while if the manufacturer were to use the SCAAT implementation, $m_\sigma = 3$ for each 3-axis electromagnetic response to a single excitation. For an image-based landmark tracker such as [41] the measurement function would, given estimates of the camera pose and a single landmark location, transform the landmark into camera space and then project it onto the camera image plane. In this case $m_\sigma = 2$ for the 2D image coordinates of the landmark.

The $m_\sigma \times 1$ *measurement noise vector* $\hat{\nu}_\sigma(t)$ in (8) is a normally-distributed zero-mean sequence that represents any random error (e.g. electrical noise) in the measurement. This parameter can be determined from component design specifications, and (or) confirmed by off-line measurement. For our simulations we did both. The corresponding $m_\sigma \times m_\sigma$ *measurement noise covariance matrix* is given by

$$E\{\hat{\nu}_\sigma(t)\hat{\nu}_\sigma^T(t+\varepsilon)\} = \begin{cases} R_\sigma(t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases} . \qquad (9)$$

For each measurement function $\hat{h}_\sigma(\bullet)$ we determine the corresponding Jacobian function

$$H_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t)[i,j] \equiv \frac{\partial}{\partial \hat{x}[j]} \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t)[i] , \qquad (10)$$

where $1 \le i \le m_\sigma$ and $1 \le j \le n$. Finally, we note the use of the standard (Kalman filter) $n \times n$ *error covariance matrix* $P(t)$ which maintains the covariance of the error in the estimated state.

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 1	Welch 1997

3.1.2 Tracking Algorithm

Given an initial state estimate $\hat{x}(0)$ and error covariance estimate $P(0)$, the SCAAT algorithm proceeds similarly to a conventional EKF, cycling through the following steps whenever a discrete measurement $\check{z}_{\sigma,t}$ from some sensor (type σ) and source becomes available at time t:

a. Compute the time δt since the previous estimate.

b. Predict the state and error covariance.

$$\begin{aligned} \hat{x}^- &= A(\delta t)\hat{x}(t-\delta t) \\ P^- &= A(\delta t)P(t-\delta t)A^T(\delta t) + Q(\delta t) \end{aligned} \tag{11}$$

c. Predict the measurement and compute the corresponding Jacobian.

$$\begin{aligned} \hat{z} &= \check{h}_\sigma(\hat{x}^-, \check{b}_t, \vec{c}_t) \\ H &= H_\sigma(\hat{x}^-, \check{b}_t, \check{e}_t) \end{aligned} \tag{12}$$

d. Compute the *Kalman gain*.

$$K = P^- H^T (HP^- H^T + R_\sigma(t))^{-1} \tag{13}$$

e. Compute the *residual* between the actual sensor measurement $\check{z}_{\sigma,t}$ and the predicted measurement from (12).

$$\vec{\Delta z} = \check{z}_{\sigma,t} - \hat{z} \tag{14}$$

f. Correct the predicted tracker state estimate and error covariance from (11).

$$\begin{aligned} \hat{x}(t) &= \hat{x}^- + K\vec{\Delta z} \\ P(t) &= (I - KH)P^- \end{aligned} \tag{15}$$

Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 1	Welch 1997
	g. Update the external orientation of equation (4) per the change indicated by the (ϕ, θ, ψ) elements of the state.* $\Delta\hat{\alpha} = \text{quaternion}(\hat{x}[\phi], \hat{x}[\theta], \hat{x}[\psi])$ $\hat{\alpha} = \hat{\alpha} \otimes \Delta\hat{\alpha}$ (16) h. Zero the orientation elements of the state vector. $\hat{x}[\phi] = \hat{x}[\theta] = \hat{x}[\psi] = 0$ (17) The equations (11)-(17) may seem computationally complex, however they can be performed quite efficiently. The computations can be optimized to eliminate operations on matrix and vector elements that are known to be zero. For example, the elements of the Jacobian H in (12) that correspond to the velocities in the state $\hat{x}(t)$ will always be zero. In addition, the matrix inverted in the computation of K in (13) is of rank m_σ (2×2 for our example in section 3.4) which is smaller for a SCAAT filter than for a corresponding conventional EKF implementation. Finally, the increased data rate allows the use of the *small angle approximations* $\sin(\theta) = \theta$ and $\cos(\theta) = 1$ in $\check{h}_\sigma(\bullet)$ and $H_\sigma(\bullet)$. The total *per estimate* computation time can therefore actually be less than that of a corresponding conventional implementation. (We are able to execute the SCAAT filter computations, with the autocalibration computations discussed in the next section, in approximately 100µs on a 200 MHz PC-compatible computer.) Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 1	Welch 1997

3.1.3 Discussion

The key to the SCAAT method is the number of constraints provided by the measurement vector and measurement function in equation (8). For the 3D-tracking problem being solved, a unique solution requires $C = 6$ non-degenerate constraints to resolve six degrees of freedom. Because individual sensor measurements typically provide less than six constraints, conventional implementations usually construct a complete measurement vector

$$\hat{z}_t = \left[\hat{z}^T_{\sigma_1, t_1} \cdots \hat{z}^T_{\sigma_N, t_N} \right]^T$$

from some group of $N \geq C$ individual sensor measurements over time $t_1 \ldots t_N$, and *then* proceed to compute an estimate. Or a particular implementation may operate in a *moving-window* fashion, combining the most recent measurement with the $N-1$ previous measurements, possibly implementing a form of a finite-impulse-response filter. In any case, for such well-constrained systems complete observability is obtained at each step of the filter. Systems that collect measurements sequentially in this way inherently violate the simultaneity assumption, as well as increase the time δt between estimates.

Welch 1997 at Section 3.1.3.

In contrast, the SCAAT method blends individual measurements that each provide incomplete constraints into a complete state estimate. The EKF inherently provides the means for this blending, no matter how complete the information content of each individual measurement $\hat{z}_{\sigma, t}$. The EKF accomplishes this through the Kalman gain K which is computed in (13). The Kalman gain, which is used to adjust the state and the error covariance in (15), is optimal in the sense that it minimizes the error covariance if certain conditions are met. Note that the inversion in (13) forms a ratio that reflects the relative uncertainties of the state and the measurement. Note too that the ratio is affected by the use of the measurement function Jacobian H. Because the Jacobian reflects the rate of change of each measurement with respect to the current state, it indicates a direction in state space along which a measurement could *possibly* affect the state. Because the gain is recomputed at each step with the appropriate measurement function and associated Jacobian, it inherently reflects the amount and direction of information provided by the individual constraint.

Welch 1997 at Section 3.1.3.

Exhibit E-7

CLAIM 1	Welch 1997

3.2 Autocalibration

The method we use for autocalibration involves *augmenting* the *main tracker filter* presented in section 3.1 to effectively implement a distinct *device filter*, a Kalman filter, for each source or sensor to be calibrated. (We use the word "device" here to include for example scene landmarks which can be thought of as passive sources, and cameras which are indeed sensors.) In general, any constant device-related parameters used by a measurement function $\hat{h}_\sigma(\bullet)$ from (8) are candidates for this autocalibration method. We assume that the parameters to be estimated are contained in the device parameter vectors $\hat{b}_t$ and $\hat{c}_t$, and we also present the case where both the source and sensor are to be calibrated since omission of one or the other is trivial. We note the following new convention.

$\widehat{\alpha}$ = augmented matrix/vector (wide hat)

Welch 1997 at Section 3.2.

3.2.1 Device Filters

For each device (source, sensor, landmark, etc.) we create a distinct device filter as follows. Let $\hat{\pi}$ represent the corresponding device parameter vector and $n_\pi = \text{length}(\hat{\pi})$.

a. Allocate an $n_\pi \times 1$ state vector $\hat{x}_\pi$ for the device, initialize with the best *a priori* device parameter estimates, e.g. from design.

b. Allocate an $n_\pi \times n_\pi$ noise covariance matrix $Q_\pi(\delta t)$, initialize with the expected parameter variances.

c. Allocate an $n_\pi \times n_\pi$ error covariance matrix $P_\pi(t)$, initialize to indicate the level of confidence in the *a priori* device parameter estimates from (a) above.

Welch 1997 at Section 3.2.1.

Exhibit E-7

CLAIM 1	Welch 1997

3.2.2 Revised Tracking Algorithm

The algorithm for tracking with concurrent autocalibration is the same as that presented in section 3.1, with the following exceptions. After step (a) in the original algorithm, we form augmented versions of the state vector

$$\hat{x}(t-\delta t) = \left[\hat{x}^T(t-\delta t)\ \ \hat{x}_{b,t}^T(t-\delta t)\ \hat{x}_{c,t}^T(t-\delta t)\right]^T, \quad (18)$$

the error covariance matrix

$$\hat{P}(t-\delta t) = \begin{bmatrix} P(t-\delta t) & 0 & 0 \\ 0 & P_{b,t}(t-\delta t) & 0 \\ 0 & 0 & P_{c,t}(t-\delta t) \end{bmatrix}, \quad (19)$$

the state transition matrix

$$\hat{A}(\delta t) = \begin{bmatrix} A(\delta t) & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}, \quad (20)$$

and the process noise matrix

$$\hat{Q}(\delta t) = \begin{bmatrix} Q(\delta t) & 0 & 0 \\ 0 & Q_{b,t}(\delta t) & 0 \\ 0 & 0 & Q_{c,t}(\delta t) \end{bmatrix}. \quad (21)$$

Welch 1997 at Section 3.2.2.

Exhibit E-7

CLAIM 1	Welch 1997
	We then follow steps (b)-(h) from the original algorithm, making the appropriate substitutions of (18)-(21), and noting that the measurement and Jacobian functions used in step (c) have become $\hat{h}_\sigma(\widehat{x}(t))$ and $H_\sigma(\widehat{x}(t))$ because the estimates of parameters $\hat{b}_t$ and $\hat{\partial}_t$ ($\hat{x}_{b,t}$ and $\hat{x}_{c,t}$) are now contained in the augmented state vector $\widehat{x}$ per (18). After step (h) we finish by extracting and saving the device filter portions of the augmented state vector and error covariance matrix $\hat{x}_{b,t}(t) = \widehat{x}(t)[i\ldots j]$ $P_{b,t}(t) = \widehat{P}(t)[i\ldots j, i\ldots j]$ (22) $\hat{x}_{c,t}(t) = \widehat{x}(t)[k\ldots l]$ $P_{c,t}(t) = \widehat{P}(t)[k\ldots l, k\ldots l]$ where $i = n+1$ $j = n+n_b$ $k = n+n_b+1$ $l = n+n_b+n_c$ Welch 1997 at Section 3.2.2.

Exhibit E-7

CLAIM 1	Welch 1997
	and n, n_b, and n_c are the dimensions of the state vectors for the main tracker filter, the source filter, and the sensor filter (respectively). We leave the main tracker filter state vector and error covariance matrix in their augmented counterparts, while we swap the device filter components in and out with each estimate. The result is that individual device filters are updated less frequently than the main tracker filter. The more a device is used, the more it is calibrated. If a device is never used, it is never calibrated. With respect to added time complexity, the computations can again be optimized to eliminate operations on matrix and vector elements that are known to be zero: those places mentioned in section 3.1, and see (19)-(21). Also note that the size of and thus time for the matrix inversion in (13) has not changed. With respect to added space complexity, the autocalibration method requires storing a separate state vector and covariance matrix for each device—a fixed amount of (generally small) space per device. For example, consider autocalibrating the beacon (LED) positions for an optical tracking system with 3,000 beacons. For each beacon one would need 3 words for the beacon state (its 3D position), $3 \times 3 = 9$ words for the noise covariance matrix, and $3 \times 3 = 9$ words for the error covariance matrix. Assuming 8 bytes per word, this is only $3{,}000 \times 8 \times (3 + 9 + 9) = 504{,}000$ bytes. Welch 1997 at Section 3.2.2.

Exhibit E-7

CLAIM 1	Welch 1997
	 Figure 3: The HiBall is shown here with the internal circuitry exposed and the lenses removed. The sensors, which can be seen through the lens openings, are mounted on PC boards that fold-up into the HiBall upon assembly. The mechanical pencil at the bottom conveys an indication of the relative size of the unit. Welch 1997 at Figure 3.

Exhibit E-7

CLAIM 1	Welch 1997
	4.1 Tracker Filter The 12 element state vector $\hat{x}(t)$ for the main tracker filter contained the elements shown in (3). Each of the 3000 beacon filters was allocated a 3 element state vector $$\hat{x}_b = \left[x_b\ y_b\ z_b\right]^T$$ where (x_b, y_b, z_b) represents the beacon's estimated position in cartesian (world) coordinates. The 12×12 state transition matrix for the main tracker filter was formed as discussed section 3.1, and for each beacon filter it was the 3×3 identity matrix. The 12×12 process noise matrix for the main tracker was computed using (7), using elements of $\hat{\eta}$ that were determined off-line using Powell's method and a variety of real motion data. For each beacon filter we used an identical noise covariance matrix $$Q_b(\delta t)[i,j] = \begin{cases} \eta_b & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$ for $1 \le i, j \le 3$, with beacon position variance η_b also determined off-line. (See [47] for the complete details.) At each estimate step, the *augmented* 15 element state vector, 15×15 process noise matrix, 15×15 state transition matrix, and 15×15 error covariance matrix all resembled (18)-(21) (without the camera parameter components). The measurement noise model was distance dependent (beacon light falls-off with distance) so $R_\sigma(t)$ from (9) was computed prior to step (d), by using a beacon distance estimate (obtained from the user and beacon positions in the predicted state $\hat{x}^-$) to project a distance-dependent electrical variance onto the camera. Welch 1997 at Section 4.1. *See* Disclosures with respect to Element 1.b; *see also* Defendants' Invalidity Contentions for further discussion.

B. DEPENDENT CLAIM 2

CLAIM 2	Welch 1997
[2] The system of claim 1 wherein the sensor	At least under Plaintiffs' apparent infringement theory, Welch 1997 discloses, either expressly or inherently, the system of claim 1 wherein the sensor subsystem includes one or more sensor modules, each providing an interface

CLAIM 2	Welch 1997
subsystem includes one or more sensor modules, each providing an interface for interacting with a corresponding set of one or more sensing elements.	for interacting with a corresponding set of one or more sensing elements. In the alternative, this element would be obvious over Welch 1997 in light of the other references disclosed in Defendants' Invalidity Contentions and/or the knowledge of one of ordinary skill in the art. *See, e.g.*: **Tracking systems determine the user's pose by measuring signals from low-level hardware sensors.** For reasons of physics and economics, most systems make multiple sequential measurements which are then combined to produce a single tracker report. For example, commercial magnetic trackers using the SPASYN (Space Synchro) system sequentially measure three magnetic vectors and then combine them mathematically to produce a report of the sensor pose. Welch 1997 at Abstract. **Our new approach produces tracker reports as each new low-level sensor measurement is made rather than waiting to form a complete collection of observations.** Because single observations under-constrain the mathematical solution, we refer to our approach as single-constraint-at-a-time or SCAAT tracking. **The key is that the single observations provide some information about the user's state, and thus can be used to incrementally improve a previous estimate. We recursively apply this principle, incorporating new sensor data as soon as it is measured. With this approach we are able to generate estimates more frequently, with less latency, and with improved accuracy. We present results from both an actual implementation, and from extensive simulations.** Welch 1997 at Abstract. **The method facilitates user motion prediction, multisensor or multiple modality data fusion, and in systems where the constraints can only be determined sequentially, it provides estimates at a higher rate and with lower latency than multiple-constraint (batch) approaches.** Welch 1997 at Section 1.

Exhibit E-7

CLAIM 2	Welch 1997
	1.1 Incomplete Information The idea that one might build a tracking system that generates a new estimate with each individual sensor measurement or *observation* is a very interesting one. After all, individual observations usually provide only partial information about a user's complete state (pose), i.e. they are "incomplete" observations. For example, for a camera observing landmarks in a scene, only limited information is obtained from observations of any single landmark. In terms of control theory, a system designed to operate with only such incomplete measurements is characterized as *unobservable* because the user state cannot be observed (determined) from the measurements. The notion of observability can also be described in terms of constraints on the unknown parameters of the system being estimated, e.g. constraints on the unknown elements of the system state. Given a particular system, and the corresponding set of unknowns that are to be estimated, let C be defined as the minimal number of independent simultaneous constraints necessary to uniquely determine a solution, let N be the number actually used to generate a new estimate, and let N_{ind} be the number of *independent* constraints that can be formed from the N constraints. For any $N \geq N_{ind}$ constraints, if $N_{ind} = C$ the problem is *well constrained*, if $N_{ind} > C$ it is *over constrained*, and if $N_{ind} < C$ it is *under-constrained*. (See Figure 1.) Welch 1997 at Section 1.1.

Exhibit E-7

CLAIM 2	Welch 1997

1.2 Landmark Tracking

Consider for example a system in which a single camera is used to observe known scene points to determine the camera position and orientation. In this case, the constraints provided by the observations are multi-dimensional: 2D image coordinates of 3D scene points. Given the internal camera parameters, a set of four known coplanar scene points, and the corresponding image coordinates, the camera position and orientation can be uniquely determined in closed-form [16]. In other words if $N = C = 4$ constraints (2D image points) are used to estimate the camera position and orientation, the system is completely observable. On the other hand, if $N < C$ then there are multiple solutions. For example with only $N = 3$ non-collinear points, there are up to 4 solutions. Even worse, with $N = 2$ or $N = 1$ points, there are infinite combinations of position and orientation that could result in the same camera images.

Welch 1997 at Section 1.2.

In general, for closed-form tracking approaches, a well or over-constrained system with $N \geq C$ is observable, an under-constrained system with $N < C$ is not. Therefore, if the individual observations provide only partial information, i.e. the measurements provide insufficient constraints, then multiple devices or landmarks must be excited and (or) sensed prior to estimating a solution. Sometimes the necessary observations can be obtained simultaneously, and sometimes they can not. Magnetic trackers such as those made by Polhemus and Ascension perform three *sequential* source excitations, each in conjunction with a complete sensor unit observation. And while a camera can indeed observe multiple landmarks simultaneously in a single image, the image processing to identify and locate the individual landmarks must be done sequentially for a single CPU system. If the landmarks can move independently over time, for example if they are artificial marks placed on the skin of an ultrasound patient for the purpose of landmark-based tracking [41], batch processing of the landmarks can reduce the effectiveness of the system. A SCAAT implementation might grab an image, extract a *single* landmark, update the estimates of both the camera *and* landmark positions, and then throw-away the image. In this way estimates are generated faster and with the most recent landmark configurations.

Welch 1997 at Section 1.2.

Exhibit E-7

CLAIM 2	Welch 1997

Figure 3: The HiBall is shown here with the internal circuitry exposed and the lenses removed. The sensors, which can be seen through the lens openings, are mounted on PC boards that fold-up into the HiBall upon assembly. The mechanical pencil at the bottom conveys an indication of the relative size of the unit.

Welch 1997 at Figure 3.

1.3 Putting the Pieces Together

Given a tracker that uses multiple constraints that are each individually incomplete, a *measurement model* for any one of incomplete constraints would be characterized as *locally unobservable*. Such a system must incorporate a sufficient set of these incomplete constraints so that the resulting overall system is observable. The corresponding aggregate measurement model can then be characterized as *globally observable*. Global observability can be obtained over *space* or over *time*. The SCAAT method adopts the latter scheme, even in some cases where the former is possible.

Welch 1997 at Section 1.3.

Exhibit E-7

CLAIM 2	Welch 1997
	The SCAAT method employs a Kalman filter (KF) in an unusual fashion. The Kalman filter is a mathematical procedure that provides an efficient computational (recursive) method for the least-squares estimation of a linear system. **It does so in a predictor-corrector fashion, predicting short-term (since the last estimate) changes in the state using a dynamic model, and then correcting them with a measurement and a corresponding measurement model. The extended Kalman filter (EKF) is a variation of the Kalman filter that supports estimation of nonlinear systems, e.g. 3D position and orientation tracking systems.** A basic introduction to the Kalman filter can be found in Chapter 1 of [31], while a more complete introductory discussion can be found in [40], which also contains some interesting historical narrative. More extensive references can be found in [7,18,24,28,31,46]. Welch 1997 at Section 3. The Kalman filter has been employed previously for virtual environment tracking estimation and prediction. For example see [2,5,12,14,42], and most recently [32]. In each of these cases however the filter was applied directly and only to the 6D pose estimates delivered by the off-the-shelf tracker. **The SCAAT approach could be applied to either a hybrid system using off-the shelf and/or custom trackers, or it could be employed by tracker developers to improve the existing systems for the end-user graphics community.** Welch 1997 at Section 3. In this section we describe the method in a manner that does not imply a specific tracking system. (In section 3.4 we present experimental results of a specific implementation, a SCAAT wide area optoelectronic tracking system.) In section 3.1 we describe the method for tracking, and in section 3.2 we describe one possible method for concurrent autocalibration. Welch 1997 at Section 3.

Exhibit E-7

CLAIM 2	Welch 1997

Throughout we use the following conventions.

x = scalar (lower case)

$\vec{x}$ = general vector (lower case, arrow) indexed as $\vec{x}[r]$

$\hat{x}$ = filter estimate vector (lower case, hat)

A = matrix (capital letters) indexed as $A[r, c]$

A^{-1} = matrix inverse

I = the identity matrix

β^{-} = matrix/vector *prediction* (super minus)

β^{T} = matrix/vector transpose (super T)

α_i = matrix/vector/scalar identifier (subscript)

$E\{\bullet\}$ = mathematical expectation

Welch 1997 at Section 3.

3.1 Tracking

3.1.1 Main Tracker Filter

The use of a Kalman filter requires a mathematical (state-space) model for the dynamics of the process to be estimated, the target motion in this case. While several possible dynamic models and associated state configurations are possible, we have found a simple *position-velocity* model to suffice for the dynamics of our applications. In fact we use this same form of model, with different parameters, for all six of the position and orientation components $(x, y, z, \phi, \theta, \psi)$. Discussion of some other potential models and the associated trade-offs can be found in [7] pp. 415-420. Because our implementation is discrete with inter sample time δt we model the target's dynamic motion with the following linear difference equation:

$$\vec{x}(t+\delta t) = A(\delta t)\vec{x}(t) + \vec{w}(\delta t). \qquad (2)$$

Welch 1997 at Section 3.1.

Exhibit E-7

CLAIM 2	Welch 1997

In the standard model corresponding to equation (2), the n dimensional Kalman filter *state vector* $\hat{x}(t)$ would completely describe the target position and orientation at any time t. In practice we use a method similar to [2,6] and maintain the complete target orientation externally to the Kalman filter in order to avoid the nonlinearities associated with orientation computations. In the internal state vector $\hat{x}(t)$ we maintain the target position as the Cartesian coordinates (x, y, z), and the *incremental* orientation as small rotations (ϕ, θ, ψ) about the (x, y, z) axis. Externally we maintain the target orientation as the *external quaternion* $\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z))$. (See [9] for discussion of quaternions.) At each filter update step, the incremental orientations (ϕ, θ, ψ) are factored into the external quaternion $\hat{\alpha}$, and then zeroed as shown below. Thus the incremental orientations are linearized for the EKF, centered about zero. We maintain the derivatives of the target position and orientation internally, in the state vector $\hat{x}(t)$. We maintain the angular velocities internally because the angular velocities behave like orthogonal vectors and do not exhibit the nonlinearities of the angles themselves. The target state is then represented by the $n = 12$ element internal state vector

$$\hat{x} = \left[x \; y \; z \; \dot{x} \; \dot{y} \; \dot{z} \; \phi \; \theta \; \psi \; \dot{\phi} \; \dot{\theta} \; \dot{\psi} \right]^T \qquad (3)$$

and the four-element external orientation quaternion

$$\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z)), \qquad (4)$$

where the time designations have been omitted for clarity.

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 2	Welch 1997

The $n \times n$ *state transition matrix* $A(\delta t)$ in (2) projects the state forward from time t to time $t + \delta t$. For our linear model, the matrix implements the relationships

$$\begin{aligned} x(t+\delta t) &= x(t) + \dot{x}(t)\delta t \\ \dot{x}(t+\delta t) &= \dot{x}(t) \end{aligned} \qquad (5)$$

and likewise for the remaining elements of (3).

The $n \times 1$ *process noise vector* $\vec{w}(\delta t)$ in (2) is a normally-distributed zero-mean sequence that represents the uncertainty in the target state over any time interval δt. The corresponding $n \times n$ *process noise covariance matrix* is given by

$$E\{\vec{w}(\delta t)\vec{w}^T(\delta t + \varepsilon)\} = \begin{cases} Q(\delta t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \qquad (6)$$

Because our implementation is discrete with inter sample time δt, we can use the transfer function method illustrated by [7] pp. 221-222 to compute a *sampled* process noise covariance matrix. (Because the associated random processes are presumed to be time stationary, we present the process noise covariance matrix as a function of the inter-sample duration δt only.) The non-zero elements of $Q(\delta t)$ are given by

$$\begin{aligned} Q(\delta t)[i,i] &= \vec{\eta}[i]\frac{(\delta t)^3}{3} \\ Q(\delta t)[i,j] = Q(\delta t)[j,i] &= \vec{\eta}[i]\frac{(\delta t)^2}{2} \\ Q(\delta t)[j,j] &= \vec{\eta}[i](\delta t) \end{aligned} \qquad (7)$$

for each pair

$$(i,j) \in \{(x,\dot{x}), (y,\dot{y}), (z,\dot{z}), (\phi,\dot{\phi}), (\theta,\dot{\theta}), (\psi,\dot{\psi})\}.$$

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 2	Welch 1997

The $\hat{\eta}[i]$ in (7) are the *correlation kernels* of the (assumed constant) noise sources presumed to be driving the dynamic model. We determined a set of values using Powell's method, and then used these in both simulation and our real implementation. The values can be "tuned" for different dynamics, though we have found that the tracker works well over a broad range of values.

The use of a Kalman filter requires not only a dynamic model as described above, but also a *measurement model* for each available type of measurement. The measurement model is used to predict the ideal noise-free response of each sensor and source pair, given the filter's current estimate of the target state as in equations (3) and (4).

It is the nature of the measurement models and indeed the actual sensor measurements that distinguishes a SCAAT Kalman filter from a well-constrained one.

For each sensor *type* σ we define the $m_\sigma \times 1$ *measurement vector* $\hat{z}_\sigma(t)$ and corresponding *measurement function* $\hat{h}_\sigma(\bullet)$ such that

$$\hat{z}_{\sigma,t} = \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t) + \hat{v}_\sigma(t) . \qquad (8)$$

Note that in the "purest" SCAAT implementation $m_\sigma = 1$ and the measurements are incorporated as single scalar values. However if it is not possible or necessary to isolate the measurements, e.g. to perform autocalibration, then multi-dimensional measurements can be incorporated also. Guidelines presented in [47] lead to the following heuristic for choosing the SCAAT Kalman filter measurement elements (constraints):

During each SCAAT Kalman filter measurement update one should observe a single sensor and source pair only.

For example, to incorporate magnetic tracker data as an end-user, $m_\sigma = 7$ for the three position and four orientation (quaternion)

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 2	Welch 1997

elements, while if the manufacturer were to use the SCAAT implementation, $m_\sigma = 3$ for each 3-axis electromagnetic response to a single excitation. For an image-based landmark tracker such as [41] the measurement function would, given estimates of the camera pose and a single landmark location, transform the landmark into camera space and then project it onto the camera image plane. In this case $m_\sigma = 2$ for the 2D image coordinates of the landmark.

The $m_\sigma \times 1$ *measurement noise vector* $\hat{\nu}_\sigma(t)$ in (8) is a normally-distributed zero-mean sequence that represents any random error (e.g. electrical noise) in the measurement. This parameter can be determined from component design specifications, and (or) confirmed by off-line measurement. For our simulations we did both. The corresponding $m_\sigma \times m_\sigma$ *measurement noise covariance matrix* is given by

$$E\{\hat{\nu}_\sigma(t)\hat{\nu}_\sigma^T(t+\varepsilon)\} = \begin{cases} R_\sigma(t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \qquad (9)$$

For each measurement function $\hat{h}_\sigma(\bullet)$ we determine the corresponding Jacobian function

$$H_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t)[i,j] \equiv \frac{\partial}{\partial \hat{x}[j]} \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t)[i], \qquad (10)$$

where $1 \le i \le m_\sigma$ and $1 \le j \le n$. Finally, we note the use of the standard (Kalman filter) $n \times n$ *error covariance matrix* $P(t)$ which maintains the covariance of the error in the estimated state.

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 2	Welch 1997

3.1.2 Tracking Algorithm

Given an initial state estimate $\hat{x}(0)$ and error covariance estimate $P(0)$, the SCAAT algorithm proceeds similarly to a conventional EKF, cycling through the following steps whenever a discrete measurement $\vec{z}_{\sigma,t}$ from some sensor (type σ) and source becomes available at time t:

a. Compute the time δt since the previous estimate.

b. Predict the state and error covariance.

$$\begin{aligned} \hat{x}^- &= A(\delta t)\hat{x}(t-\delta t) \\ P^- &= A(\delta t)P(t-\delta t)A^T(\delta t) + Q(\delta t) \end{aligned} \tag{11}$$

c. Predict the measurement and compute the corresponding Jacobian.

$$\begin{aligned} \hat{z} &= \vec{h}_\sigma(\hat{x}^-, \vec{b}_t, \vec{c}_t) \\ H &= H_\sigma(\hat{x}^-, \vec{b}_t, \vec{e}_t) \end{aligned} \tag{12}$$

d. Compute the *Kalman gain*.

$$K = P^-H^T(HP^-H^T + R_\sigma(t))^{-1} \tag{13}$$

e. Compute the *residual* between the actual sensor measurement $\vec{z}_{\sigma,t}$ and the predicted measurement from (12).

$$\vec{\Delta z} = \vec{z}_{\sigma,t} - \hat{z} \tag{14}$$

f. Correct the predicted tracker state estimate and error covariance from (11).

$$\begin{aligned} \hat{x}(t) &= \hat{x}^- + K\vec{\Delta z} \\ P(t) &= (I-KH)P^- \end{aligned} \tag{15}$$

Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 2	Welch 1997
	g. Update the external orientation of equation (4) per the change indicated by the (ϕ, θ, ψ) elements of the state.* $\Delta\hat{\alpha} = \text{quaternion}(\hat{x}[\phi], \hat{x}[\theta], \hat{x}[\psi])$ $\hat{\alpha} = \hat{\alpha} \otimes \Delta\hat{\alpha}$ (16) h. Zero the orientation elements of the state vector. $\hat{x}[\phi] = \hat{x}[\theta] = \hat{x}[\psi] = 0$ (17) The equations (11)-(17) may seem computationally complex, however they can be performed quite efficiently. The computations can be optimized to eliminate operations on matrix and vector elements that are known to be zero. For example, the elements of the Jacobian H in (12) that correspond to the velocities in the state $\hat{x}(t)$ will always be zero. In addition, the matrix inverted in the computation of K in (13) is of rank m_σ (2×2 for our example in section 3.4) which is smaller for a SCAAT filter than for a corresponding conventional EKF implementation. Finally, the increased data rate allows the use of the *small angle approximations* $\sin(\theta) = \theta$ and $\cos(\theta) = 1$ in $\check{h}_\sigma(\bullet)$ and $H_\sigma(\bullet)$. The total *per estimate* computation time can therefore actually be less than that of a corresponding conventional implementation. (We are able to execute the SCAAT filter computations, with the autocalibration computations discussed in the next section, in approximately 100µs on a 200 MHz PC-compatible computer.) Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 2	Welch 1997

3.1.3 Discussion

The key to the SCAAT method is the number of constraints provided by the measurement vector and measurement function in equation (8). For the 3D-tracking problem being solved, a unique solution requires $C = 6$ non-degenerate constraints to resolve six degrees of freedom. Because individual sensor measurements typically provide less than six constraints, conventional implementations usually construct a complete measurement vector

$$\hat{z}_t = \left[\hat{z}^T_{\sigma_1, t_1} \cdots \hat{z}^T_{\sigma_N, t_N} \right]^T$$

from some group of $N \geq C$ individual sensor measurements over time $t_1 \ldots t_N$, and *then* proceed to compute an estimate. Or a particular implementation may operate in a *moving-window* fashion, combining the most recent measurement with the $N-1$ previous measurements, possibly implementing a form of a finite-impulse-response filter. In any case, for such well-constrained systems complete observability is obtained at each step of the filter. Systems that collect measurements sequentially in this way inherently violate the simultaneity assumption, as well as increase the time δt between estimates.

Welch 1997 at Section 3.1.3.

In contrast, the SCAAT method blends individual measurements that each provide incomplete constraints into a complete state estimate. The EKF inherently provides the means for this blending, no matter how complete the information content of each individual measurement $\hat{z}_{\sigma, t}$. The EKF accomplishes this through the Kalman gain K which is computed in (13). The Kalman gain, which is used to adjust the state and the error covariance in (15), is optimal in the sense that it minimizes the error covariance if certain conditions are met. Note that the inversion in (13) forms a ratio that reflects the relative uncertainties of the state and the measurement. Note too that the ratio is affected by the use of the measurement function Jacobian H. Because the Jacobian reflects the rate of change of each measurement with respect to the current state, it indicates a direction in state space along which a measurement could *possibly* affect the state. Because the gain is recomputed at each step with the appropriate measurement function and associated Jacobian, it inherently reflects the amount and direction of information provided by the individual constraint.

Welch 1997 at Section 3.1.3.

Exhibit E-7

CLAIM 2	Welch 1997

3.2.1 Device Filters

For each device (source, sensor, landmark, etc.) we create a distinct device filter as follows. Let $\vec{\pi}$ represent the corresponding device parameter vector and n_π = length($\vec{\pi}$).

a. Allocate an $n_\pi \times 1$ state vector $\hat{x}_\pi$ for the device, initialize with the best *a priori* device parameter estimates, e.g. from design.

b. Allocate an $n_\pi \times n_\pi$ noise covariance matrix $Q_\pi(\delta t)$, initialize with the expected parameter variances.

c. Allocate an $n_\pi \times n_\pi$ error covariance matrix $P_\pi(t)$, initialize to indicate the level of confidence in the *a priori* device parameter estimates from (a) above.

Welch 1997 at Section 3.2.1.

3.3 Stability

Because the SCAAT method uses individual measurements with insufficient information, one might be concerned about the potential for instability or divergence. A linear system is said to be stable if its response to any input tends to a finite steady value after the input is removed [24]. For the Kalman filter in general this is certainly not a new concern, and there are standard requirements and corresponding tests that ensure or detect stability (see [18], p. 132):

a. The filter must be uniformly completely observable,
b. the dynamic and measurement noise matrices in equations (6) and (9) must be bounded from above and below, and
c. the dynamic behavior represented by in equation (2) must be bounded from above.

As it turns out, these conditions and their standard tests are equally applicable to a SCAAT implementation. For the SCAAT method the conditions mean that the user dynamics between estimates must be bounded, the measurement noise must be bounded, one must incorporate a sufficient set of non-degenerate constraints over time. In particular, the constraints must be incorporated in less than 1/2 the time of the user motion time-constant in order to avoid tracking an alias of the true motion. In general these conditions are easily met for systems and circumstances that would otherwise be stable with a multiple-constraint implementation. A complete stability

Exhibit E-7

CLAIM 2	Welch 1997
	analysis is beyond the scope of this paper, and is presented in [47]. Welch 1997 at Section 3.3. [47] Greg Welch, 1996. "SCAAT: Incremental Tracking with Incomplete Information," University of North Carolina at Chapel Hill, doctoral dissertation, TR 96-051. Welch 1997 at References. **4 EXPERIMENTS** We are using the SCAAT approach in the current version of the UNC wide-area optoelectronic tracking system known as the *HiBall tracker*. The *HiBall*, shown below in Figure 3, incorporates six optical sensors and six lenses with infrared filters into one golf ball sized sensing unit that can be worn on a user's head or hand. The principal mechanical component of the HiBall, the senor housing unit, was fabricated by researchers at the University of Utah using their α1 modeling environment. Because the HiBall sensors and lenses share a common transparent space in the center of the housing, a single sensor can actually sense light through more than one lens. By making use of all of these *views* we end up with effectively 26 "cameras". These cameras are then used to observe ceiling-mounted light-emitting diodes (LEDs) to track the position and orientation of the HiBall. This inside-looking-out approach was first used with the previous UNC optoelectronic tracking system [44] which spanned most of the user's head and weighed approximately ten pounds, not including a backpack containing some electronics. In contrast, the HiBall sensing unit is the size of a golf ball and weighs only five ounces, *including* the electronics. The combination of reduced weight, smaller packaging, and the new SCAAT algorithm results in a very ergonomic, fast, and accurate system. In this section we present results from both simulations performed during the design and development of the HiBall, and Welch 1997 at Section 4.

Exhibit E-7

CLAIM 2	Welch 1997

preliminary results from the actual implementation. The simulations are useful because we have control over the "truth" and can perform controlled experiments. The results from the actual implementation serve to demonstrate actual operation and to provide some support for our accuracy and stability claims.

With respect to the SCAAT implementation, the tracker *sensors* are the HiBall cameras and the tracker *sources* are the ceiling-mounted 2D array of approximately 3000 electronic beacons (LEDs). The cameras provide a single 2D measurement vector, i.e. a 2D constraint, which is the (u, v) image coordinates of the beacon as seen by the camera. So for this example, $m_\sigma = 2$ and $\hat{z}_\sigma = [u, v]^T$. The measurement function $\hat{h}_\sigma(\bullet)$ transforms the beacon into camera coordinates and then projects it onto the camera's image plane in order to predict the camera response.

For the simulations we generated individual measurement events (a single beacon activation followed by a single camera reading) at a rate of 1000 Hz, and corrupted the measurements using the noise models detailed in [8]. We tested components of our real system in a laboratory and found the noise models in [8] to be reasonably accurate, if not pessimistic. We also perturbed the 3D beacon positions prior to simulations with a normally-distributed noise source with approximately 1.7 millimeters standard deviation. We controlled all random number generators to facilitate method comparisons with common random sequences.

Welch 1997 at Section 4.

To evaluate the filter performance we needed some reference data. Our solution was to collect motion data from *real-user* sessions with a conventional tracking system, and then to filter the data to remove high frequency noise. We then *defined* this data to be the "truth". We did this for seven such sessions.

The simulator operated by sampling the truth data, choosing one beacon and camera (round-robin from within the set of valid combinations), computing the corresponding camera measurement vector $\hat{z}_{\sigma,t}$, and then adding some measurement noise. The (noisy) measurement vector, the camera parameter vector $\hat{c}_t$ (position and orientation in user coordinates), and the beacon parameter vector $\hat{b}_t$ (position in world coordinates) were then sent to the tracker.

Welch 1997 at Section 4.

Exhibit E-7

CLAIM 2	Welch 1997
	For the tracking algorithm, we simulated both the SCAAT method (section 3.1, modified per section 3.2 for autocalibration) and several multiple-constraint methods, including the Collinearity method [3] and several variations of moving window (finite impulse response) methods. For each of the methods we varied the measurement noise, the measurement frequency, and the beacon position error. For the multiple constraint methods we also varied the number of constraints (beacon observations) per estimate N. In each case the respective estimates were compared with the truth data set for performance evaluation. Welch 1997 at Section 4. *See* Disclosures with respect to Claim 1, *supra*; *see also* Defendants' Invalidity Contentions for further discussion.

C. DEPENDENT CLAIM 3

CLAIM 3	Welch 1997
[3] The system of claim 2 wherein the interface enables the sensor module to perform computations independently of an implementation of the estimation subsystem.	At least under Plaintiffs' apparent infringement theory, Welch 1997 discloses, either expressly or inherently, the system of claim 2 wherein the interface enables the sensor module to perform computations independently of an implementation of the estimation subsystem. In the alternative, this element would be obvious over Welch 1997 in light of the other references disclosed in Defendants' Invalidity Contentions and/or the knowledge of one of ordinary skill in the art. *See, e.g.*: We present a promising new mathematical method for tracking a user's pose (position and orientation) for interactive computer graphics. The method, which is applicable to a wide variety of both commercial and experimental systems, improves accuracy by properly assimilating sequential observations, filtering sensor measurements, and by concurrently autocalibrating source and sensor devices. It facilitates user motion prediction, multisensory data fusion, and higher report rates with lower latency than previous methods.

Exhibit E-7

CLAIM 3	Welch 1997
	Tracking systems determine the user's pose by measuring signals from low-level hardware sensors. For reasons of physics and economics, most systems make multiple sequential measurements which are then combined to produce a single tracker report. For example, commercial magnetic trackers using the SPASYN (*Space Synchro*) system sequentially measure three magnetic vectors and then combine them mathematically to produce a report of the sensor pose. Our new approach produces tracker reports as each new low-level sensor measurement is made rather than waiting to form a complete collection of observations. Because single observations under-constrain the mathematical solution, we refer to our approach as single-constraint-at-a-time or SCAAT tracking. The key is that the single observations provide some information about the user's state, and thus can be used to incrementally improve a previous estimate. We recursively apply this principle, incorporating new sensor data as soon as it is measured. With this approach we are able to generate estimates more frequently, with less latency, and with improved accuracy. We present results from both an actual implementation, and from extensive simulations. Welch 1997 at Abstract. 2.2 Device Isolation & Autocalibration Knowledge about source and sensor imperfections can be used to improve the accuracy of tracking systems. While intrinsic sensor parameters can often be determined off-line, e.g. by the manufacturer, this is generally not the case for extrinsic parameters. For example it can be difficult to determine the exact geometric relationship between the various sensors of a hybrid system. Consider that the coordinate system of a magnetic sensor is located at some unknown location inside the sensor unit. Similarly the precise geometric relationship between visible landmarks used in a vision-based system is often difficult to determine. Even worse, landmark positions can change over time as, for example, a patient's skin deforms with pressure from an ultrasound probe. In general, goals such as flexibility, ease of use, and lower cost, make the notion of self-calibration or autocalibration attractive. The general idea for autocalibration is not new. See for example [19,45]. **However, because the SCAAT method isolates the measurements provided by each sensor or modality, the method provides a new and elegant means to autocalibrate concurrently while tracking. Because the SCAAT method isolates the individual measurements, or measurement dimensions, individual source and sensor imperfections are more easily identified and dealt with. Furthermore, because the simultaneity assumption is avoided, the motion restrictions discussed in section 2.1 would be removed, and autocalibration could be performed while concurrently tracking a target.**

Exhibit E-7

CLAIM 3	Welch 1997
	The isolation enforced by the SCAAT approach can improve results even if the constraints are obtained simultaneously through multidimensional measurements. An intuitive explanation is that if the elements (dimensions) are corrupted by independent noise, then incorporating the elements independently can offer improved filtering over a batch or ensemble estimation scheme. Welch 1997 at Section 2.2. **3.1 Tracking** **3.1.1 Main Tracker Filter** The use of a Kalman filter requires a mathematical (state-space) model for the dynamics of the process to be estimated, the target motion in this case. While several possible dynamic models and associated state configurations are possible, we have found a simple *position-velocity* model to suffice for the dynamics of our applications. In fact we use this same form of model, with different parameters, for all six of the position and orientation components $(x, y, z, \phi, \theta, \psi)$. Discussion of some other potential models and the associated trade-offs can be found in [7] pp. 415-420. Because our implementation is discrete with inter sample time δt we model the target's dynamic motion with the following linear difference equation: $$\hat{x}(t+\delta t) = A(\delta t)\hat{x}(t) + \hat{w}(\delta t). \quad (2)$$ Welch 1997 at Section 3.1.

Exhibit E-7

CLAIM 3	Welch 1997
	In the standard model corresponding to equation (2), the n dimensional Kalman filter *state vector* $\hat{x}(t)$ would completely describe the target position and orientation at any time t. In practice we use a method similar to [2,6] and maintain the complete target orientation externally to the Kalman filter in order to avoid the nonlinearities associated with orientation computations. In the internal state vector $\hat{x}(t)$ we maintain the target position as the Cartesian coordinates (x, y, z), and the *incremental* orientation as small rotations (ϕ, θ, ψ) about the (x, y, z) axis. Externally we maintain the target orientation as the *external quaternion* $\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z))$. (See [9] for discussion of quaternions.) At each filter update step, the incremental orientations (ϕ, θ, ψ) are factored into the external quaternion $\hat{\alpha}$, and then zeroed as shown below. Thus the incremental orientations are linearized for the EKF, centered about zero. We maintain the derivatives of the target position and orientation internally, in the state vector $\hat{x}(t)$. We maintain the angular velocities internally because the angular velocities behave like orthogonal vectors and do not exhibit the nonlinearities of the angles themselves. The target state is then represented by the $n = 12$ element internal state vector $$\hat{x} = \left[x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \phi\ \theta\ \psi\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}\right]^T \quad (3)$$ and the four-element external orientation quaternion $$\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z)), \quad (4)$$ where the time designations have been omitted for clarity. Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 3	Welch 1997

The $n \times n$ *state transition matrix* $A(\delta t)$ in (2) projects the state forward from time t to time $t + \delta t$. For our linear model, the matrix implements the relationships

$$\begin{aligned} x(t+\delta t) &= x(t) + \dot{x}(t)\delta t \\ \dot{x}(t+\delta t) &= \dot{x}(t) \end{aligned} \qquad (5)$$

and likewise for the remaining elements of (3).

The $n \times 1$ *process noise vector* $\vec{w}(\delta t)$ in (2) is a normally-distributed zero-mean sequence that represents the uncertainty in the target state over any time interval δt. The corresponding $n \times n$ *process noise covariance matrix* is given by

$$E\{\vec{w}(\delta t)\vec{w}^T(\delta t + \varepsilon)\} = \begin{cases} Q(\delta t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \qquad (6)$$

Because our implementation is discrete with inter sample time δt, we can use the transfer function method illustrated by [7] pp. 221-222 to compute a *sampled* process noise covariance matrix. (Because the associated random processes are presumed to be time stationary, we present the process noise covariance matrix as a function of the inter-sample duration δt only.) The non-zero elements of $Q(\delta t)$ are given by

$$\begin{aligned} Q(\delta t)[i,i] &= \vec{\eta}[i]\frac{(\delta t)^3}{3} \\ Q(\delta t)[i,j] = Q(\delta t)[j,i] &= \vec{\eta}[i]\frac{(\delta t)^2}{2} \\ Q(\delta t)[j,j] &= \vec{\eta}[i](\delta t) \end{aligned} \qquad (7)$$

for each pair

$$(i,j) \in \{(x,\dot{x}), (y,\dot{y}), (z,\dot{z}), (\phi,\dot{\phi}), (\theta,\dot{\theta}), (\psi,\dot{\psi})\}.$$

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 3	Welch 1997

The $\hat{\eta}[i]$ in (7) are the *correlation kernels* of the (assumed constant) noise sources presumed to be driving the dynamic model. We determined a set of values using Powell's method, and then used these in both simulation and our real implementation. The values can be "tuned" for different dynamics, though we have found that the tracker works well over a broad range of values.

The use of a Kalman filter requires not only a dynamic model as described above, but also a *measurement model* for each available type of measurement. The measurement model is used to predict the ideal noise-free response of each sensor and source pair, given the filter's current estimate of the target state as in equations (3) and (4).

It is the nature of the measurement models and indeed the actual sensor measurements that distinguishes a SCAAT Kalman filter from a well-constrained one.

For each sensor *type* σ we define the $m_\sigma \times 1$ *measurement vector* $\hat{z}_\sigma(t)$ and corresponding *measurement function* $\hat{h}_\sigma(\bullet)$ such that

$$\hat{z}_{\sigma,t} = \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t) + \hat{v}_\sigma(t)\,. \qquad (8)$$

Note that in the "purest" SCAAT implementation $m_\sigma = 1$ and the measurements are incorporated as single scalar values. However if it is not possible or necessary to isolate the measurements, e.g. to perform autocalibration, then multi-dimensional measurements can be incorporated also. Guidelines presented in [47] lead to the following heuristic for choosing the SCAAT Kalman filter measurement elements (constraints):

During each SCAAT Kalman filter measurement update one should observe a single sensor and source pair only.

For example, to incorporate magnetic tracker data as an end-user, $m_\sigma = 7$ for the three position and four orientation (quaternion)

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 3	Welch 1997

elements, while if the manufacturer were to use the SCAAT implementation, $m_\sigma = 3$ for each 3-axis electromagnetic response to a single excitation. For an image-based landmark tracker such as [41] the measurement function would, given estimates of the camera pose and a single landmark location, transform the landmark into camera space and then project it onto the camera image plane. In this case $m_\sigma = 2$ for the 2D image coordinates of the landmark.

The $m_\sigma \times 1$ *measurement noise vector* $\hat{\nu}_\sigma(t)$ in (8) is a normally-distributed zero-mean sequence that represents any random error (e.g. electrical noise) in the measurement. This parameter can be determined from component design specifications, and (or) confirmed by off-line measurement. For our simulations we did both. The corresponding $m_\sigma \times m_\sigma$ *measurement noise covariance matrix* is given by

$$E\{\hat{\nu}_\sigma(t)\hat{\nu}_\sigma^T(t+\varepsilon)\} = \begin{cases} R_\sigma(t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \qquad (9)$$

For each measurement function $\hat{h}_\sigma(\bullet)$ we determine the corresponding Jacobian function

$$H_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t)[i,j] \equiv \frac{\partial}{\partial \hat{x}[j]} \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t)[i], \qquad (10)$$

where $1 \le i \le m_\sigma$ and $1 \le j \le n$. Finally, we note the use of the standard (Kalman filter) $n \times n$ *error covariance matrix* $P(t)$ which maintains the covariance of the error in the estimated state.

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 3	Welch 1997

3.1.2 Tracking Algorithm

Given an initial state estimate $\hat{x}(0)$ and error covariance estimate $P(0)$, the SCAAT algorithm proceeds similarly to a conventional EKF, cycling through the following steps whenever a discrete measurement $\hat{z}_{\sigma,t}$ from some sensor (type σ) and source becomes available at time t:

a. Compute the time δt since the previous estimate.

b. Predict the state and error covariance.

$$\begin{aligned} \hat{x}^- &= A(\delta t)\hat{x}(t-\delta t) \\ P^- &= A(\delta t)P(t-\delta t)A^T(\delta t) + Q(\delta t) \end{aligned} \tag{11}$$

c. Predict the measurement and compute the corresponding Jacobian.

$$\begin{aligned} \hat{z} &= \hat{h}_\sigma(\hat{x}^-, \hat{b}_t, \vec{c}_t) \\ H &= H_\sigma(\hat{x}^-, \hat{b}_t, \hat{s}_t) \end{aligned} \tag{12}$$

d. Compute the *Kalman gain*.

$$K = P^-H^T(HP^-H^T + R_\sigma(t))^{-1} \tag{13}$$

e. Compute the *residual* between the actual sensor measurement $\hat{z}_{\sigma,t}$ and the predicted measurement from (12).

$$\vec{\Delta z} = \hat{z}_{\sigma,t} - \hat{z} \tag{14}$$

f. Correct the predicted tracker state estimate and error covariance from (11).

$$\begin{aligned} \hat{x}(t) &= \hat{x}^- + K\vec{\Delta z} \\ P(t) &= (I-KH)P^- \end{aligned} \tag{15}$$

Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 3	Welch 1997
	g. Update the external orientation of equation (4) per the change indicated by the (ϕ, θ, ψ) elements of the state. $\Delta\hat{\alpha} = \text{quaternion}(\hat{x}[\phi], \hat{x}[\theta], \hat{x}[\psi])$ $\hat{\alpha} = \hat{\alpha} \otimes \Delta\hat{\alpha}$ (16) h. Zero the orientation elements of the state vector. $\hat{x}[\phi] = \hat{x}[\theta] = \hat{x}[\psi] = 0$ (17) The equations (11)-(17) may seem computationally complex, however they can be performed quite efficiently. The computations can be optimized to eliminate operations on matrix and vector elements that are known to be zero. For example, the elements of the Jacobian H in (12) that correspond to the velocities in the state $\hat{x}(t)$ will always be zero. In addition, the matrix inverted in the computation of K in (13) is of rank m_σ (2×2 for our example in section 3.4) which is smaller for a SCAAT filter than for a corresponding conventional EKF implementation. Finally, the increased data rate allows the use of the *small angle approximations* $\sin(\theta) = \theta$ and $\cos(\theta) = 1$ in $\hat{h}_\sigma(\bullet)$ and $H_\sigma(\bullet)$. The total *per estimate* computation time can therefore actually be less than that of a corresponding conventional implementation. (We are able to execute the SCAAT filter computations, with the autocalibration computations discussed in the next section, in approximately 100µs on a 200 MHz PC-compatible computer.) Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 3	Welch 1997

3.1.3 Discussion

The key to the SCAAT method is the number of constraints provided by the measurement vector and measurement function in equation (8). For the 3D-tracking problem being solved, a unique solution requires $C = 6$ non-degenerate constraints to resolve six degrees of freedom. Because individual sensor measurements typically provide less than six constraints, conventional implementations usually construct a complete measurement vector

$$\hat{z}_t = \left[\hat{z}^T_{\sigma_1, t_1} \cdots \hat{z}^T_{\sigma_N, t_N} \right]^T$$

from some group of $N \geq C$ individual sensor measurements over time $t_1 \ldots t_N$, and *then* proceed to compute an estimate. Or a particular implementation may operate in a *moving-window* fashion, combining the most recent measurement with the $N-1$ previous measurements, possibly implementing a form of a finite-impulse-response filter. In any case, for such well-constrained systems complete observability is obtained at each step of the filter. Systems that collect measurements sequentially in this way inherently violate the simultaneity assumption, as well as increase the time δt between estimates.

In contrast, the SCAAT method blends individual measurements that each provide incomplete constraints into a complete state estimate. The EKF inherently provides the means for this blending, no matter how complete the information content of each individual measurement $\hat{z}_{\sigma, t}$. The EKF accomplishes this through the Kalman gain K which is computed in (13). The Kalman gain, which is used to adjust the state and the error covariance in (15), is optimal in the sense that it minimizes the error covariance if certain conditions are met. Note that the inversion in (13) forms a ratio that reflects the relative uncertainties of the state and the measurement. Note too that the ratio is affected by the use of the measurement function Jacobian H. Because the Jacobian reflects the rate of change of each measurement with respect to the current state, it indicates a direction in state space along which a measurement could *possibly* affect the state. Because the gain is recomputed at each step with the appropriate measurement function and associated Jacobian, it inherently reflects the amount and direction of information provided by the individual constraint.

Welch 1997 at 3.1.3.

Exhibit E-7

CLAIM 3	Welch 1997

3.2 Autocalibration

The method we use for autocalibration involves *augmenting* the *main tracker filter* presented in section 3.1 to effectively implement a distinct *device filter*, a Kalman filter, for each source or sensor to be calibrated. (We use the word "device" here to include for example scene landmarks which can be thought of as passive sources, and cameras which are indeed sensors.) In general, any constant device-related parameters used by a measurement function $\check{h}_\sigma(\bullet)$ from (8) are candidates for this autocalibration method. We assume that the parameters to be estimated are contained in the device parameter vectors $\check{b}_t$ and $\check{c}_t$, and we also present the case where both the source and sensor are to be calibrated since omission of one or the other is trivial. We note the following new convention.

$\widehat{\alpha}$ = augmented matrix/vector (wide hat)

Welch 1997 at Section 3.2

3.2.1 Device Filters

For each device (source, sensor, landmark, etc.) we create a distinct device filter as follows. Let $\check{\pi}$ represent the corresponding device parameter vector and n_π = length($\check{\pi}$).

a. Allocate an $n_\pi \times 1$ state vector $\hat{x}_\pi$ for the device, initialize with the best *a priori* device parameter estimates, e.g. from design.

b. Allocate an $n_\pi \times n_\pi$ noise covariance matrix $Q_\pi(\delta t)$, initialize with the expected parameter variances.

c. Allocate an $n_\pi \times n_\pi$ error covariance matrix $P_\pi(t)$, initialize to indicate the level of confidence in the *a priori* device parameter estimates from (a) above.

Welch 1997 at Section 3.2.1.

Exhibit E-7

CLAIM 3	Welch 1997

3.2.2 Revised Tracking Algorithm

The algorithm for tracking with concurrent autocalibration is the same as that presented in section 3.1, with the following exceptions. After step (a) in the original algorithm, we form augmented versions of the state vector

$$\hat{\mathbf{x}}(t-\delta t) = \left[\hat{\mathbf{x}}^T(t-\delta t)\ \hat{\mathbf{x}}_{b,t}^T(t-\delta t)\ \hat{\mathbf{x}}_{c,t}^T(t-\delta t)\right]^T, \quad (18)$$

the error covariance matrix

$$\hat{\mathbf{P}}(t-\delta t) = \begin{bmatrix} \mathbf{P}(t-\delta t) & 0 & 0 \\ 0 & \mathbf{P}_{b,t}(t-\delta t) & 0 \\ 0 & 0 & \mathbf{P}_{c,t}(t-\delta t) \end{bmatrix}, \quad (19)$$

the state transition matrix

$$\hat{\mathbf{A}}(\delta t) = \begin{bmatrix} \mathbf{A}(\delta t) & 0 & 0 \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I} \end{bmatrix}, \quad (20)$$

and the process noise matrix

$$\hat{\mathbf{Q}}(\delta t) = \begin{bmatrix} \mathbf{Q}(\delta t) & 0 & 0 \\ 0 & \mathbf{Q}_{b,t}(\delta t) & 0 \\ 0 & 0 & \mathbf{Q}_{c,t}(\delta t) \end{bmatrix}. \quad (21)$$

Welch 1997 at Section 3.2.2.

Exhibit E-7

CLAIM 3	Welch 1997

We then follow steps (b)-(h) from the original algorithm, making the appropriate substitutions of (18)-(21), and noting that the measurement and Jacobian functions used in step (c) have become $\check{h}_\sigma(\widehat{x}(t))$ and $H_\sigma(\widehat{x}(t))$ because the estimates of parameters $\check{b}_t$ and $\check{\alpha}_t$ ($\hat{x}_{b,t}$ and $\hat{x}_{c,t}$) are now contained in the augmented state vector $\widehat{x}$ per (18). After step (h) we finish by extracting and saving the device filter portions of the augmented state vector and error covariance matrix

$$\begin{aligned} \hat{x}_{b,t}(t) &= \widehat{x}(t)[i..j] \\ P_{b,t}(t) &= \widehat{P}(t)[i..j, i..j] \\ \hat{x}_{c,t}(t) &= \widehat{x}(t)[k..l] \\ P_{c,t}(t) &= \widehat{P}(t)[k..l, k..l] \end{aligned} \tag{22}$$

where

$$\begin{aligned} i &= n+1 \\ j &= n+n_b \\ k &= n+n_b+1 \\ l &= n+n_b+n_c \end{aligned}$$

and n, n_b, and n_c are the dimensions of the state vectors for the main tracker filter, the source filter, and the sensor filter (respectively). We leave the main tracker filter state vector and error covariance matrix in their augmented counterparts, while we swap the device filter components in and out with each estimate. The result is that individual device filters are updated less frequently than the main tracker filter. The more a device is used, the more it is calibrated. If a device is never used, it is never calibrated.

Welch 1997 at Section 3.2.2.

Exhibit E-7

CLAIM 3	Welch 1997

With respect to added time complexity, the computations can again be optimized to eliminate operations on matrix and vector elements that are known to be zero: those places mentioned in section 3.1, and see (19)-(21). Also note that the size of and thus time for the matrix inversion in (13) has not changed. With respect to added space complexity, the autocalibration method requires storing a separate state vector and covariance matrix for each device—a fixed amount of (generally small) space per device. For example, consider autocalibrating the beacon (LED) positions for an optical tracking system with 3,000 beacons. For each beacon one would need 3 words for the beacon state (its 3D position), $3 \times 3 = 9$ words for the noise covariance matrix, and $3 \times 3 = 9$ words for the error covariance matrix. Assuming 8 bytes per word, this is only $3{,}000 \times 8 \times (3 + 9 + 9) = 504{,}000$ bytes.

Welch 1997 at Section 3.2.2.

3.2.3 Discussion

The ability to simultaneously estimate two dependent sets of unknowns (the target and device states) is made possible by several factors. First, the dynamics of the two sets are very different as would be reflected in the process noise matrices. We assume the target is undergoing some random (constant) acceleration, reflected in the noise parameter η of $Q(\delta t)$ in (6). Conversely, we assume the device parameters are constant, and so the elements of $Q_\pi(\delta t)$ for a source or sensor simply reflect any allowed variances in the corresponding parameters: usually zero or extremely small. In addition, while the target is expected to be moving, the filter expects the motion between any two estimations to closely correspond to the velocity estimates in the state (3). If the tracker estimate rate is high enough, poorly estimated device parameters will result in what appears to be almost instantaneous target motion. The increased rate of the SCAAT method allows such motion to be recognized as unlikely, and attributed to poorly estimated device parameters.

Welch 1997 at Section 3.2.3

Exhibit E-7

CLAIM 3	Welch 1997
	4 EXPERIMENTS We are using the SCAAT approach in the current version of the UNC wide-area optoelectronic tracking system known as the *HiBall tracker*. The *HiBall*, shown below in Figure 3, incorporates six optical sensors and six lenses with infrared filters into one golf ball sized sensing unit that can be worn on a user's head or hand. The principal mechanical component of the HiBall, the senor housing unit, was fabricated by researchers at the University of Utah using their α1 modeling environment. Because the HiBall sensors and lenses share a common transparent space in the center of the housing, a single sensor can actually sense light through more than one lens. By making use of all of these *views* we end up with effectively 26 "cameras". These cameras are then used to observe ceiling-mounted light-emitting diodes (LEDs) to track the position and orientation of the HiBall. This inside-looking-out approach was first used with the previous UNC optoelectronic tracking system [44] which spanned most of the user's head and weighed approximately ten pounds, not including a backpack containing some electronics. In contrast, the HiBall sensing unit is the size of a golf ball and weighs only five ounces, *including* the electronics. The combination of reduced weight, smaller packaging, and the new SCAAT algorithm results in a very ergonomic, fast, and accurate system. In this section we present results from both simulations performed during the design and development of the HiBall, and Welch 1997 at Section 4.

Exhibit E-7

CLAIM 3	Welch 1997

preliminary results from the actual implementation. The simulations are useful because we have control over the "truth" and can perform controlled experiments. The results from the actual implementation serve to demonstrate actual operation and to provide some support for our accuracy and stability claims.

With respect to the SCAAT implementation, the tracker *sensors* are the HiBall cameras and the tracker *sources* are the ceiling-mounted 2D array of approximately 3000 electronic beacons (LEDs). The cameras provide a single 2D measurement vector, i.e. a 2D constraint, which is the (u, v) image coordinates of the beacon as seen by the camera. So for this example, $m_\sigma = 2$ and $\hat{z}_\sigma = [u, v]^T$. The measurement function $\hat{h}_\sigma(\bullet)$ transforms the beacon into camera coordinates and then projects it onto the camera's image plane in order to predict the camera response.

For the simulations we generated individual measurement events (a single beacon activation followed by a single camera reading) at a rate of 1000 Hz, and corrupted the measurements using the noise models detailed in [8]. We tested components of our real system in a laboratory and found the noise models in [8] to be reasonably accurate, if not pessimistic. We also perturbed the 3D beacon positions prior to simulations with a normally-distributed noise source with approximately 1.7 millimeters standard deviation. We controlled all random number generators to facilitate method comparisons with common random sequences.

Welch 1997 at Section 4.

To evaluate the filter performance we needed some reference data. Our solution was to collect motion data from *real-user* sessions with a conventional tracking system, and then to filter the data to remove high frequency noise. We then *defined* this data to be the "truth". We did this for seven such sessions.

The simulator operated by sampling the truth data, choosing one beacon and camera (round-robin from within the set of valid combinations), computing the corresponding camera measurement vector $\hat{z}_{\sigma,t}$, and then adding some measurement noise. The (noisy) measurement vector, the camera parameter vector $\hat{c}_t$ (position and orientation in user coordinates), and the beacon parameter vector $\hat{b}_t$ (position in world coordinates) were then sent to the tracker.

Welch 1997 at Section 4.

Exhibit E-7

CLAIM 3	Welch 1997

For the tracking algorithm, we simulated both the SCAAT method (section 3.1, modified per section 3.2 for autocalibration) and several multiple-constraint methods, including the Collinearity method [3] and several variations of moving window (finite impulse response) methods. For each of the methods we varied the measurement noise, the measurement frequency, and the beacon position error. For the multiple constraint methods we also varied the number of constraints (beacon observations) per estimate N. In each case the respective estimates were compared with the truth data set for performance evaluation.

Welch 1997 at Section 4.

Figure 3: The HiBall is shown here with the internal circuitry exposed and the lenses removed. The sensors, which can be seen through the lens openings, are mounted on PC boards that fold-up into the HiBall upon assembly. The mechanical pencil at the bottom conveys an indication of the relative size of the unit.

Welch 1997 at Figure 3.

CLAIM 3	Welch 1997
	See Disclosures with respect to Claim 2, *supra*; *see also* Defendants' Invalidity Contentions for further discussion.

D. DEPENDENT CLAIM 4

CLAIM 4	Welch 1997
[4] The system of claim 2 wherein the interface enables the estimation subsystem to perform computations independently of an implementation of the sensor modules.	At least under Plaintiffs' apparent infringement theory, Welch 1997 discloses, either expressly or inherently, the system of claim 2 wherein the interface enables the estimation subsystem to perform computations independently of an implementation of the sensor modules. In the alternative, this element would be obvious over Welch 1997 in light of the other references disclosed in Defendants' Invalidity Contentions and/or the knowledge of one of ordinary skill in the art. *See, e.g.*: We present a promising new mathematical method for tracking a user's pose (position and orientation) for interactive computer graphics. The method, which is applicable to a wide variety of both commercial and experimental systems, improves accuracy by properly assimilating sequential observations, filtering sensor measurements, and by concurrently autocalibrating source and sensor devices. It facilitates user motion prediction, multisensory data fusion, and higher report rates with lower latency than previous methods. Tracking systems determine the user's pose by measuring signals from low-level hardware sensors. For reasons of physics and economics, most systems make multiple sequential measurements which are then combined to produce a single tracker report. For example, commercial magnetic trackers using the SPASYN (*Space Synchro*) system sequentially measure three magnetic vectors and then combine them mathematically to produce a report of the sensor pose. Our new approach produces tracker reports as each new low-level sensor measurement is made rather than waiting to form a complete collection of observations. Because single observations under-constrain the mathematical solution, we refer to our approach as single-constraint-at-a-time or SCAAT tracking. The key is that the single observations provide some information about the user's state, and thus can be used to incrementally improve a previous estimate. We recursively apply this principle, incorporating new sensor data as soon as it is measured. With this approach we are able to generate estimates more frequently, with less latency, and with improved

CLAIM 4	Welch 1997
	accuracy. We present results from both an actual implementation, and from extensive simulations. Welch 1997 at Abstract. 2.2 Device Isolation & Autocalibration Knowledge about source and sensor imperfections can be used to improve the accuracy of tracking systems. While intrinsic sensor parameters can often be determined off-line, e.g. by the manufacturer, this is generally not the case for extrinsic parameters. For example it can be difficult to determine the exact geometric relationship between the various sensors of a hybrid system. Consider that the coordinate system of a magnetic sensor is located at some unknown location inside the sensor unit. Similarly the precise geometric relationship between visible landmarks used in a vision-based system is often difficult to determine. Even worse, landmark positions can change over time as, for example, a patient's skin deforms with pressure from an ultrasound probe. In general, goals such as flexibility, ease of use, and lower cost, make the notion of self-calibration or autocalibration attractive. The general idea for autocalibration is not new. See for example [19,45]. **However, because the SCAAT method isolates the measurements provided by each sensor or modality, the method provides a new and elegant means to autocalibrate concurrently while tracking. Because the SCAAT method isolates the individual measurements, or measurement dimensions, individual source and sensor imperfections are more easily identified and dealt with. Furthermore, because the simultaneity assumption is avoided, the motion restrictions discussed in section 2.1 would be removed, and autocalibration could be performed while concurrently tracking a target.** The isolation enforced by the SCAAT approach can improve results even if the constraints are obtained simultaneously through multidimensional measurements. An intuitive explanation is that if the elements (dimensions) are corrupted by independent noise, then incorporating the elements independently can offer improved filtering over a batch or ensemble estimation scheme. Welch 1997 at Section 2.2.

Exhibit E-7

CLAIM 4	Welch 1997
	3.1 Tracking **3.1.1 Main Tracker Filter** The use of a Kalman filter requires a mathematical (state-space) model for the dynamics of the process to be estimated, the target motion in this case. While several possible dynamic models and associated state configurations are possible, we have found a simple *position-velocity* model to suffice for the dynamics of our applications. In fact we use this same form of model, with different parameters, for all six of the position and orientation components $(x, y, z, \phi, \theta, \psi)$. Discussion of some other potential models and the associated trade-offs can be found in [7] pp. 415-420. Because our implementation is discrete with inter sample time δt we model the target's dynamic motion with the following linear difference equation: $$\hat{x}(t+\delta t) = A(\delta t)\hat{x}(t) + \hat{w}(\delta t). \quad (2)$$ Welch 1997 at Section 3.1.

Exhibit E-7

CLAIM 4	Welch 1997
	In the standard model corresponding to equation (2), the n dimensional Kalman filter *state vector* $\hat{x}(t)$ would completely describe the target position and orientation at any time t. In practice we use a method similar to [2,6] and maintain the complete target orientation externally to the Kalman filter in order to avoid the nonlinearities associated with orientation computations. In the internal state vector $\hat{x}(t)$ we maintain the target position as the Cartesian coordinates (x, y, z), and the *incremental* orientation as small rotations (ϕ, θ, ψ) about the (x, y, z) axis. Externally we maintain the target orientation as the *external quaternion* $\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z))$. (See [9] for discussion of quaternions.) At each filter update step, the incremental orientations (ϕ, θ, ψ) are factored into the external quaternion $\hat{\alpha}$, and then zeroed as shown below. Thus the incremental orientations are linearized for the EKF, centered about zero. We maintain the derivatives of the target position and orientation internally, in the state vector $\hat{x}(t)$. We maintain the angular velocities internally because the angular velocities behave like orthogonal vectors and do not exhibit the nonlinearities of the angles themselves. The target state is then represented by the $n = 12$ element internal state vector $$\hat{x} = \left[x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \phi\ \theta\ \psi\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}\right]^T \quad (3)$$ and the four-element external orientation quaternion $$\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z)), \quad (4)$$ where the time designations have been omitted for clarity. Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 4	Welch 1997

The $n \times n$ *state transition matrix* $A(\delta t)$ in (2) projects the state forward from time t to time $t + \delta t$. For our linear model, the matrix implements the relationships

$$\begin{aligned} x(t+\delta t) &= x(t) + \dot{x}(t)\delta t \\ \dot{x}(t+\delta t) &= \dot{x}(t) \end{aligned} \qquad (5)$$

and likewise for the remaining elements of (3).

The $n \times 1$ *process noise vector* $\vec{w}(\delta t)$ in (2) is a normally-distributed zero-mean sequence that represents the uncertainty in the target state over any time interval δt. The corresponding $n \times n$ *process noise covariance matrix* is given by

$$E\{\vec{w}(\delta t)\vec{w}^T(\delta t + \varepsilon)\} = \begin{cases} Q(\delta t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \qquad (6)$$

Because our implementation is discrete with inter sample time δt, we can use the transfer function method illustrated by [7] pp. 221-222 to compute a *sampled* process noise covariance matrix. (Because the associated random processes are presumed to be time stationary, we present the process noise covariance matrix as a function of the inter-sample duration δt only.) The non-zero elements of $Q(\delta t)$ are given by

$$\begin{aligned} Q(\delta t)[i,i] &= \vec{\eta}[i]\frac{(\delta t)^3}{3} \\ Q(\delta t)[i,j] = Q(\delta t)[j,i] &= \vec{\eta}[i]\frac{(\delta t)^2}{2} \\ Q(\delta t)[j,j] &= \vec{\eta}[i](\delta t) \end{aligned} \qquad (7)$$

for each pair

$$(i,j) \in \{(x,\dot{x}), (y,\dot{y}), (z,\dot{z}), (\phi,\dot{\phi}), (\theta,\dot{\theta}), (\psi,\dot{\psi})\}.$$

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 4	Welch 1997

The $\hat{\eta}[i]$ in (7) are the *correlation kernels* of the (assumed constant) noise sources presumed to be driving the dynamic model. We determined a set of values using Powell's method, and then used these in both simulation and our real implementation. The values can be "tuned" for different dynamics, though we have found that the tracker works well over a broad range of values.

The use of a Kalman filter requires not only a dynamic model as described above, but also a *measurement model* for each available type of measurement. The measurement model is used to predict the ideal noise-free response of each sensor and source pair, given the filter's current estimate of the target state as in equations (3) and (4).

It is the nature of the measurement models and indeed the actual sensor measurements that distinguishes a SCAAT Kalman filter from a well-constrained one.

For each sensor *type* σ we define the $m_\sigma \times 1$ *measurement vector* $\hat{z}_\sigma(t)$ and corresponding *measurement function* $\hat{h}_\sigma(\bullet)$ such that

$$\hat{z}_{\sigma,t} = \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t) + \hat{v}_\sigma(t) . \qquad (8)$$

Note that in the "purest" SCAAT implementation $m_\sigma = 1$ and the measurements are incorporated as single scalar values. However if it is not possible or necessary to isolate the measurements, e.g. to perform autocalibration, then multi-dimensional measurements can be incorporated also. Guidelines presented in [47] lead to the following heuristic for choosing the SCAAT Kalman filter measurement elements (constraints):

During each SCAAT Kalman filter measurement update one should observe a single sensor and source pair only.

For example, to incorporate magnetic tracker data as an end-user, $m_\sigma = 7$ for the three position and four orientation (quaternion)

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 4	Welch 1997

elements, while if the manufacturer were to use the SCAAT implementation, $m_\sigma = 3$ for each 3-axis electromagnetic response to a single excitation. For an image-based landmark tracker such as [41] the measurement function would, given estimates of the camera pose and a single landmark location, transform the landmark into camera space and then project it onto the camera image plane. In this case $m_\sigma = 2$ for the 2D image coordinates of the landmark.

The $m_\sigma \times 1$ *measurement noise vector* $\hat{\nu}_\sigma(t)$ in (8) is a normally-distributed zero-mean sequence that represents any random error (e.g. electrical noise) in the measurement. This parameter can be determined from component design specifications, and (or) confirmed by off-line measurement. For our simulations we did both. The corresponding $m_\sigma \times m_\sigma$ *measurement noise covariance matrix* is given by

$$E\{\hat{\nu}_\sigma(t)\hat{\nu}_\sigma^T(t+\varepsilon)\} = \begin{cases} R_\sigma(t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \qquad (9)$$

For each measurement function $\hat{h}_\sigma(\bullet)$ we determine the corresponding Jacobian function

$$H_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t)[i,j] \equiv \frac{\partial}{\partial \hat{x}[j]} \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t)[i], \qquad (10)$$

where $1 \le i \le m_\sigma$ and $1 \le j \le n$. Finally, we note the use of the standard (Kalman filter) $n \times n$ *error covariance matrix* $P(t)$ which maintains the covariance of the error in the estimated state.

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 4	Welch 1997

3.1.2 Tracking Algorithm

Given an initial state estimate $\hat{x}(0)$ and error covariance estimate $P(0)$, the SCAAT algorithm proceeds similarly to a conventional EKF, cycling through the following steps whenever a discrete measurement $\hat{z}_{\sigma,t}$ from some sensor (type σ) and source becomes available at time t:

a. Compute the time δt since the previous estimate.

b. Predict the state and error covariance.

$$\begin{aligned} \hat{x}^- &= A(\delta t)\hat{x}(t-\delta t) \\ P^- &= A(\delta t)P(t-\delta t)A^T(\delta t) + Q(\delta t) \end{aligned} \tag{11}$$

c. Predict the measurement and compute the corresponding Jacobian.

$$\begin{aligned} \hat{z} &= \hat{h}_\sigma(\hat{x}^-, \hat{b}_t, \vec{c}_t) \\ H &= H_\sigma(\hat{x}^-, \hat{b}_t, \hat{s}_t) \end{aligned} \tag{12}$$

d. Compute the *Kalman gain*.

$$K = P^-H^T(HP^-H^T + R_\sigma(t))^{-1} \tag{13}$$

e. Compute the *residual* between the actual sensor measurement $\hat{z}_{\sigma,t}$ and the predicted measurement from (12).

$$\vec{\Delta z} = \hat{z}_{\sigma,t} - \hat{z} \tag{14}$$

f. Correct the predicted tracker state estimate and error covariance from (11).

$$\begin{aligned} \hat{x}(t) &= \hat{x}^- + K\vec{\Delta z} \\ P(t) &= (I-KH)P^- \end{aligned} \tag{15}$$

Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 4	Welch 1997

g. Update the external orientation of equation (4) per the change indicated by the (ϕ, θ, ψ) elements of the state.

$$\Delta\hat{\alpha} = \text{quaternion}(\hat{x}[\phi], \hat{x}[\theta], \hat{x}[\psi])$$
$$\hat{\alpha} = \hat{\alpha} \otimes \Delta\hat{\alpha} \quad (16)$$

h. Zero the orientation elements of the state vector.

$$\hat{x}[\phi] = \hat{x}[\theta] = \hat{x}[\psi] = 0 \quad (17)$$

The equations (11)-(17) may seem computationally complex, however they can be performed quite efficiently. The computations can be optimized to eliminate operations on matrix and vector elements that are known to be zero. For example, the elements of the Jacobian H in (12) that correspond to the velocities in the state $\hat{x}(t)$ will always be zero. In addition, the matrix inverted in the computation of K in (13) is of rank m_σ (2×2 for our example in section 3.4) which is smaller for a SCAAT filter than for a corresponding conventional EKF implementation. Finally, the increased data rate allows the use of the *small angle approximations* $\sin(\theta) = \theta$ and $\cos(\theta) = 1$ in $\hat{h}_\sigma(\bullet)$ and $H_\sigma(\bullet)$. The total *per estimate* computation time can therefore actually be less than that of a corresponding conventional implementation. (We are able to execute the SCAAT filter computations, with the autocalibration computations discussed in the next section, in approximately 100µs on a 200 MHz PC-compatible computer.)

Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 4	Welch 1997

3.1.3 Discussion

The key to the SCAAT method is the number of constraints provided by the measurement vector and measurement function in equation (8). For the 3D-tracking problem being solved, a unique solution requires $C = 6$ non-degenerate constraints to resolve six degrees of freedom. Because individual sensor measurements typically provide less than six constraints, conventional implementations usually construct a complete measurement vector

$$\hat{z}_t = \left[\hat{z}^T_{\sigma_1, t_1} \cdots \hat{z}^T_{\sigma_N, t_N} \right]^T$$

from some group of $N \geq C$ individual sensor measurements over time $t_1 \ldots t_N$, and *then* proceed to compute an estimate. Or a particular implementation may operate in a *moving-window* fashion, combining the most recent measurement with the $N-1$ previous measurements, possibly implementing a form of a finite-impulse-response filter. In any case, for such well-constrained systems complete observability is obtained at each step of the filter. Systems that collect measurements sequentially in this way inherently violate the simultaneity assumption, as well as increase the time δt between estimates.

In contrast, the SCAAT method blends individual measurements that each provide incomplete constraints into a complete state estimate. The EKF inherently provides the means for this blending, no matter how complete the information content of each individual measurement $\hat{z}_{\sigma, t}$. The EKF accomplishes this through the Kalman gain K which is computed in (13). The Kalman gain, which is used to adjust the state and the error covariance in (15), is optimal in the sense that it minimizes the error covariance if certain conditions are met. Note that the inversion in (13) forms a ratio that reflects the relative uncertainties of the state and the measurement. Note too that the ratio is affected by the use of the measurement function Jacobian H. Because the Jacobian reflects the rate of change of each measurement with respect to the current state, it indicates a direction in state space along which a measurement could *possibly* affect the state. Because the gain is recomputed at each step with the appropriate measurement function and associated Jacobian, it inherently reflects the amount and direction of information provided by the individual constraint.

Welch 1997 at 3.1.3.

Exhibit E-7

CLAIM 4	Welch 1997

3.2 Autocalibration

The method we use for autocalibration involves *augmenting* the *main tracker filter* presented in section 3.1 to effectively implement a distinct *device filter*, a Kalman filter, for each source or sensor to be calibrated. (We use the word "device" here to include for example scene landmarks which can be thought of as passive sources, and cameras which are indeed sensors.) In general, any constant device-related parameters used by a measurement function $\check{h}_\sigma(\bullet)$ from (8) are candidates for this autocalibration method. We assume that the parameters to be estimated are contained in the device parameter vectors $\check{b}_t$ and $\check{c}_t$, and we also present the case where both the source and sensor are to be calibrated since omission of one or the other is trivial. We note the following new convention.

$\widehat{\alpha}$ = augmented matrix/vector (wide hat)

Welch 1997 at Section 3.2

3.2.1 Device Filters

For each device (source, sensor, landmark, etc.) we create a distinct device filter as follows. Let $\check{\pi}$ represent the corresponding device parameter vector and n_π = length($\check{\pi}$).

a. Allocate an $n_\pi \times 1$ state vector $\hat{x}_\pi$ for the device, initialize with the best *a priori* device parameter estimates, e.g. from design.

b. Allocate an $n_\pi \times n_\pi$ noise covariance matrix $Q_\pi(\delta t)$, initialize with the expected parameter variances.

c. Allocate an $n_\pi \times n_\pi$ error covariance matrix $P_\pi(t)$, initialize to indicate the level of confidence in the *a priori* device parameter estimates from (a) above.

Welch 1997 at Section 3.2.1.

Exhibit E-7

CLAIM 4	Welch 1997

3.2.2 Revised Tracking Algorithm

The algorithm for tracking with concurrent autocalibration is the same as that presented in section 3.1, with the following exceptions. After step (a) in the original algorithm, we form augmented versions of the state vector

$$\hat{\mathbf{x}}(t-\delta t) = \left[\hat{\mathbf{x}}^T(t-\delta t)\ \hat{\mathbf{x}}_{b,t}^T(t-\delta t)\ \hat{\mathbf{x}}_{c,t}^T(t-\delta t)\right]^T, \quad (18)$$

the error covariance matrix

$$\hat{\mathbf{P}}(t-\delta t) = \begin{bmatrix} \mathbf{P}(t-\delta t) & 0 & 0 \\ 0 & \mathbf{P}_{b,t}(t-\delta t) & 0 \\ 0 & 0 & \mathbf{P}_{c,t}(t-\delta t) \end{bmatrix}, \quad (19)$$

the state transition matrix

$$\hat{\mathbf{A}}(\delta t) = \begin{bmatrix} \mathbf{A}(\delta t) & 0 & 0 \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I} \end{bmatrix}, \quad (20)$$

and the process noise matrix

$$\hat{\mathbf{Q}}(\delta t) = \begin{bmatrix} \mathbf{Q}(\delta t) & 0 & 0 \\ 0 & \mathbf{Q}_{b,t}(\delta t) & 0 \\ 0 & 0 & \mathbf{Q}_{c,t}(\delta t) \end{bmatrix}. \quad (21)$$

Welch 1997 at Section 3.2.2.

Exhibit E-7

CLAIM 4	Welch 1997

We then follow steps (b)-(h) from the original algorithm, making the appropriate substitutions of (18)-(21), and noting that the measurement and Jacobian functions used in step (c) have become $\check{h}_\sigma(\widehat{x}(t))$ and $H_\sigma(\widehat{x}(t))$ because the estimates of parameters $\check{b}_t$ and $\check{\alpha}_t$ ($\hat{x}_{b,t}$ and $\hat{x}_{c,t}$) are now contained in the augmented state vector $\widehat{x}$ per (18). After step (h) we finish by extracting and saving the device filter portions of the augmented state vector and error covariance matrix

$$\begin{aligned} \hat{x}_{b,t}(t) &= \widehat{x}(t)[i..j] \\ P_{b,t}(t) &= \widehat{P}(t)[i..j, i..j] \\ \hat{x}_{c,t}(t) &= \widehat{x}(t)[k..l] \\ P_{c,t}(t) &= \widehat{P}(t)[k..l, k..l] \end{aligned} \tag{22}$$

where

$$\begin{aligned} i &= n+1 \\ j &= n+n_b \\ k &= n+n_b+1 \\ l &= n+n_b+n_c \end{aligned}$$

and n, n_b, and n_c are the dimensions of the state vectors for the main tracker filter, the source filter, and the sensor filter (respectively). We leave the main tracker filter state vector and error covariance matrix in their augmented counterparts, while we swap the device filter components in and out with each estimate. The result is that individual device filters are updated less frequently than the main tracker filter. The more a device is used, the more it is calibrated. If a device is never used, it is never calibrated.

Welch 1997 at Section 3.2.2.

Exhibit E-7

CLAIM 4	Welch 1997

With respect to added time complexity, the computations can again be optimized to eliminate operations on matrix and vector elements that are known to be zero: those places mentioned in section 3.1, and see (19)-(21). Also note that the size of and thus time for the matrix inversion in (13) has not changed. With respect to added space complexity, the autocalibration method requires storing a separate state vector and covariance matrix for each device—a fixed amount of (generally small) space per device. For example, consider autocalibrating the beacon (LED) positions for an optical tracking system with 3,000 beacons. For each beacon one would need 3 words for the beacon state (its 3D position), $3 \times 3 = 9$ words for the noise covariance matrix, and $3 \times 3 = 9$ words for the error covariance matrix. Assuming 8 bytes per word, this is only $3{,}000 \times 8 \times (3 + 9 + 9) = 504{,}000$ bytes.

Welch 1997 at Section 3.2.2.

3.2.3 Discussion

The ability to simultaneously estimate two dependent sets of unknowns (the target and device states) is made possible by several factors. First, the dynamics of the two sets are very different as would be reflected in the process noise matrices. We assume the target is undergoing some random (constant) acceleration, reflected in the noise parameter η of $Q(\delta t)$ in (6). Conversely, we assume the device parameters are constant, and so the elements of $Q_\pi(\delta t)$ for a source or sensor simply reflect any allowed variances in the corresponding parameters: usually zero or extremely small. In addition, while the target is expected to be moving, the filter expects the motion between any two estimations to closely correspond to the velocity estimates in the state (3). If the tracker estimate rate is high enough, poorly estimated device parameters will result in what appears to be almost instantaneous target motion. The increased rate of the SCAAT method allows such motion to be recognized as unlikely, and attributed to poorly estimated device parameters.

Welch 1997 at Section 3.2.3

Exhibit E-7

CLAIM 4	Welch 1997
	4 EXPERIMENTS We are using the SCAAT approach in the current version of the UNC wide-area optoelectronic tracking system known as the *HiBall tracker*. The *HiBall*, shown below in Figure 3, incorporates six optical sensors and six lenses with infrared filters into one golf ball sized sensing unit that can be worn on a user's head or hand. The principal mechanical component of the HiBall, the senor housing unit, was fabricated by researchers at the University of Utah using their α1 modeling environment. Because the HiBall sensors and lenses share a common transparent space in the center of the housing, a single sensor can actually sense light through more than one lens. By making use of all of these *views* we end up with effectively 26 "cameras". These cameras are then used to observe ceiling-mounted light-emitting diodes (LEDs) to track the position and orientation of the HiBall. This inside-looking-out approach was first used with the previous UNC optoelectronic tracking system [44] which spanned most of the user's head and weighed approximately ten pounds, not including a backpack containing some electronics. In contrast, the HiBall sensing unit is the size of a golf ball and weighs only five ounces, *including* the electronics. The combination of reduced weight, smaller packaging, and the new SCAAT algorithm results in a very ergonomic, fast, and accurate system. In this section we present results from both simulations performed during the design and development of the HiBall, and Welch 1997 at Section 4.

Exhibit E-7

CLAIM 4	Welch 1997

preliminary results from the actual implementation. The simulations are useful because we have control over the "truth" and can perform controlled experiments. The results from the actual implementation serve to demonstrate actual operation and to provide some support for our accuracy and stability claims.

With respect to the SCAAT implementation, the tracker *sensors* are the HiBall cameras and the tracker *sources* are the ceiling-mounted 2D array of approximately 3000 electronic beacons (LEDs). The cameras provide a single 2D measurement vector, i.e. a 2D constraint, which is the (u, v) image coordinates of the beacon as seen by the camera. So for this example, $m_\sigma = 2$ and $\hat{z}_\sigma = [u, v]^T$. The measurement function $\hat{h}_\sigma(\bullet)$ transforms the beacon into camera coordinates and then projects it onto the camera's image plane in order to predict the camera response.

For the simulations we generated individual measurement events (a single beacon activation followed by a single camera reading) at a rate of 1000 Hz, and corrupted the measurements using the noise models detailed in [8]. We tested components of our real system in a laboratory and found the noise models in [8] to be reasonably accurate, if not pessimistic. We also perturbed the 3D beacon positions prior to simulations with a normally-distributed noise source with approximately 1.7 millimeters standard deviation. We controlled all random number generators to facilitate method comparisons with common random sequences.

Welch 1997 at Section 4.

To evaluate the filter performance we needed some reference data. Our solution was to collect motion data from *real-user* sessions with a conventional tracking system, and then to filter the data to remove high frequency noise. We then *defined* this data to be the "truth". We did this for seven such sessions.

The simulator operated by sampling the truth data, choosing one beacon and camera (round-robin from within the set of valid combinations), computing the corresponding camera measurement vector $\hat{z}_{\sigma,t}$, and then adding some measurement noise. The (noisy) measurement vector, the camera parameter vector $\hat{c}_t$ (position and orientation in user coordinates), and the beacon parameter vector $\hat{b}_t$ (position in world coordinates) were then sent to the tracker.

Welch 1997 at Section 4.

Exhibit E-7

CLAIM 4	Welch 1997

For the tracking algorithm, we simulated both the SCAAT method (section 3.1, modified per section 3.2 for autocalibration) and several multiple-constraint methods, including the Collinearity method [3] and several variations of moving window (finite impulse response) methods. For each of the methods we varied the measurement noise, the measurement frequency, and the beacon position error. For the multiple constraint methods we also varied the number of constraints (beacon observations) per estimate N. In each case the respective estimates were compared with the truth data set for performance evaluation.

Welch 1997 at Section 4.

Figure 3: The HiBall is shown here with the internal circuitry exposed and the lenses removed. The sensors, which can be seen through the lens openings, are mounted on PC boards that fold-up into the HiBall upon assembly. The mechanical pencil at the bottom conveys an indication of the relative size of the unit.

Welch 1997 at Figure 3.

CLAIM 4	Welch 1997
	See Disclosures with respect to Claim 2, *supra*; *see also* Defendants' Invalidity Contentions for further discussion.

E. INDEPENDENT CLAIM 6

CLAIM 6	Welch 1997
[6.pre] A method comprising:	At least under Plaintiffs' apparent infringement theory, Welch 1997 discloses, either expressly or inherently, a method. In the alternative, this element would be obvious over Welch 1997 in light of the other references disclosed in Defendants' Invalidity Contentions and/or the knowledge of one of ordinary skill in the art. *See, e.g.*: **We present a promising new mathematical method for tracking a user's pose (position and orientation) for interactive computer graphics. The method, which is applicable to a wide variety of both commercial and experimental systems, improves accuracy by properly assimilating sequential observations, filtering sensor measurements, and by concurrently autocalibrating source and sensor devices. It facilitates user motion prediction, multisensor data fusion, and higher report rates with lower latency than previous methods.** **Tracking systems determine the user's pose by measuring signals from low-level hardware sensors.** For reasons of physics and economics, most systems make multiple sequential measurements which are then combined to produce a single tracker report. For example, commercial magnetic trackers using the SPASYN (Space Synchro) system sequentially measure three magnetic vectors and then combine them mathematically to produce a report of the sensor pose. **Our new approach produces tracker reports as each new low-level sensor measurement is made rather than waiting to form a complete collection of observations.** Because single observations under-constrain the mathematical solution, we refer to our approach as single-constraint-at-a-time or SCAAT tracking. The key is that the single observations provide some information about the user's state, and thus can be used to incrementally improve a previous estimate. **We recursively apply this principle, incorporating new sensor data as soon as it is measured. With this approach we are able to generate estimates more frequently, with less latency, and with improved accuracy. We present results from both an actual implementation, and**

CLAIM 6	Welch 1997
	from extensive simulations. Welch 1997 at Abstract. **The method we present requires, we believe, a fundamental change in the way people think about estimating a set of unknowns in general, and tracking for virtual environments in particular.** Most of us have the preconceived notion that to estimate a set of unknowns we need as many constraints as there are degrees of freedom at any particular instant in time. What we present instead is a method to constrain the unknowns over time, continually refining an estimate for the solution, a single constraint at a time. Welch 1997 at Section 1. **The SCAAT method employs a Kalman filter (KF) in an unusual fashion. The Kalman filter is a mathematical procedure that provides an efficient computational (recursive) method for the least-squares estimation of a linear system. It does so in a predictor-corrector fashion, predicting short-term (since the last estimate) changes in the state using a dynamic model, and then correcting them with a measurement and a corresponding measurement model. The extended Kalman filter (EKF) is a variation of the Kalman filter that supports estimation of nonlinear systems, e.g. 3D position and orientation tracking systems.** A basic introduction to the Kalman filter can be found in Chapter 1 of [31], while a more complete introductory discussion can be found in [40], which also contains some interesting historical narrative. More extensive references can be found in [7,18,24,28,31,46]. The Kalman filter has been employed previously for virtual environment tracking estimation and prediction. For example see [2,5,12,14,42], and most recently [32]. In each of these cases however the filter was applied directly and only to the 6D pose estimates delivered by the off-the-shelf tracker. The SCAAT approach could be applied to either a hybrid system using off-the-shelf and/or custom trackers, or it could be employed by tracker developers to improve the existing systems for the end-user graphics community. **In this section we describe the method in a manner that does not imply a specific tracking system. (In section 3.4 we present experimental results of a specific implementation, a SCAAT wide-area optoelectronic tracking system.) In section 3.1 we describe the method for tracking, and in section 3.2 we describe one possible method for concurrent autocalibration.** Welch 1997 at Section 3.

Exhibit E-7

CLAIM 6	Welch 1997
	3.1 Tracking **3.1.1 Main Tracker Filter** The use of a Kalman filter requires a mathematical (state-space) model for the dynamics of the process to be estimated, the target motion in this case. While several possible dynamic models and associated state configurations are possible, we have found a simple *position-velocity* model to suffice for the dynamics of our applications. In fact we use this same form of model, with different parameters, for all six of the position and orientation components $(x, y, z, \phi, \theta, \psi)$. Discussion of some other potential models and the associated trade-offs can be found in [7] pp. 415-420. Because our implementation is discrete with inter sample time δt we model the target's dynamic motion with the following linear difference equation: $$\hat{x}(t+\delta t) = A(\delta t)\hat{x}(t) + \hat{w}(\delta t). \quad (2)$$ Welch 1997 at Section 3.1.

Exhibit E-7

CLAIM 6	Welch 1997
	In the standard model corresponding to equation (2), the n dimensional Kalman filter *state vector* $\hat{x}(t)$ would completely describe the target position and orientation at any time t. In practice we use a method similar to [2,6] and maintain the complete target orientation externally to the Kalman filter in order to avoid the nonlinearities associated with orientation computations. In the internal state vector $\hat{x}(t)$ we maintain the target position as the Cartesian coordinates (x, y, z), and the *incremental* orientation as small rotations (ϕ, θ, ψ) about the (x, y, z) axis. Externally we maintain the target orientation as the *external quaternion* $\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z))$. (See [9] for discussion of quaternions.) At each filter update step, the incremental orientations (ϕ, θ, ψ) are factored into the external quaternion $\hat{\alpha}$, and then zeroed as shown below. Thus the incremental orientations are linearized for the EKF, centered about zero. We maintain the derivatives of the target position and orientation internally, in the state vector $\hat{x}(t)$. We maintain the angular velocities internally because the angular velocities behave like orthogonal vectors and do not exhibit the nonlinearities of the angles themselves. The target state is then represented by the $n = 12$ element internal state vector $$\hat{x} = \left[x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \phi\ \theta\ \psi\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}\right]^T \quad (3)$$ and the four-element external orientation quaternion $$\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z)), \quad (4)$$ where the time designations have been omitted for clarity. Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 6	Welch 1997

The $n \times n$ *state transition matrix* $A(\delta t)$ in (2) projects the state forward from time t to time $t + \delta t$. For our linear model, the matrix implements the relationships

$$\begin{aligned} x(t+\delta t) &= x(t) + \dot{x}(t)\delta t \\ \dot{x}(t+\delta t) &= \dot{x}(t) \end{aligned} \qquad (5)$$

and likewise for the remaining elements of (3).

The $n \times 1$ *process noise vector* $\vec{w}(\delta t)$ in (2) is a normally-distributed zero-mean sequence that represents the uncertainty in the target state over any time interval δt. The corresponding $n \times n$ *process noise covariance matrix* is given by

$$E\{\vec{w}(\delta t)\vec{w}^T(\delta t + \varepsilon)\} = \begin{cases} Q(\delta t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \qquad (6)$$

Because our implementation is discrete with inter sample time δt, we can use the transfer function method illustrated by [7] pp. 221-222 to compute a *sampled* process noise covariance matrix. (Because the associated random processes are presumed to be time stationary, we present the process noise covariance matrix as a function of the inter-sample duration δt only.) The non-zero elements of $Q(\delta t)$ are given by

$$\begin{aligned} Q(\delta t)[i,i] &= \vec{\eta}[i]\frac{(\delta t)^3}{3} \\ Q(\delta t)[i,j] = Q(\delta t)[j,i] &= \vec{\eta}[i]\frac{(\delta t)^2}{2} \\ Q(\delta t)[j,j] &= \vec{\eta}[i](\delta t) \end{aligned} \qquad (7)$$

for each pair

$$(i,j) \in \{(x,\dot{x}), (y,\dot{y}), (z,\dot{z}), (\phi,\dot{\phi}), (\theta,\dot{\theta}), (\psi,\dot{\psi})\}.$$

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 6	Welch 1997

The $\hat{\eta}[i]$ in (7) are the *correlation kernels* of the (assumed constant) noise sources presumed to be driving the dynamic model. We determined a set of values using Powell's method, and then used these in both simulation and our real implementation. The values can be "tuned" for different dynamics, though we have found that the tracker works well over a broad range of values.

The use of a Kalman filter requires not only a dynamic model as described above, but also a *measurement model* for each available type of measurement. The measurement model is used to predict the ideal noise-free response of each sensor and source pair, given the filter's current estimate of the target state as in equations (3) and (4).

It is the nature of the measurement models and indeed the actual sensor measurements that distinguishes a SCAAT Kalman filter from a well-constrained one.

For each sensor *type* σ we define the $m_\sigma \times 1$ *measurement vector* $\hat{z}_\sigma(t)$ and corresponding *measurement function* $\hat{h}_\sigma(\bullet)$ such that

$$\hat{z}_{\sigma,t} = \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t) + \hat{v}_\sigma(t)\,. \qquad (8)$$

Note that in the "purest" SCAAT implementation $m_\sigma = 1$ and the measurements are incorporated as single scalar values. However if it is not possible or necessary to isolate the measurements, e.g. to perform autocalibration, then multi-dimensional measurements can be incorporated also. Guidelines presented in [47] lead to the following heuristic for choosing the SCAAT Kalman filter measurement elements (constraints):

During each SCAAT Kalman filter measurement update one should observe a single sensor and source pair only.

For example, to incorporate magnetic tracker data as an end-user, $m_\sigma = 7$ for the three position and four orientation (quaternion)

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 6	Welch 1997
	elements, while if the manufacturer were to use the SCAAT implementation, $m_\sigma = 3$ for each 3-axis electromagnetic response to a single excitation. For an image-based landmark tracker such as [41] the measurement function would, given estimates of the camera pose and a single landmark location, transform the landmark into camera space and then project it onto the camera image plane. In this case $m_\sigma = 2$ for the 2D image coordinates of the landmark. The $m_\sigma \times 1$ *measurement noise vector* $\hat{\nu}_\sigma(t)$ in (8) is a normally-distributed zero-mean sequence that represents any random error (e.g. electrical noise) in the measurement. This parameter can be determined from component design specifications, and (or) confirmed by off-line measurement. For our simulations we did both. The corresponding $m_\sigma \times m_\sigma$ *measurement noise covariance matrix* is given by $$E\{\hat{\nu}_\sigma(t)\hat{\nu}_\sigma^T(t+\varepsilon)\} = \begin{cases} R_\sigma(t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases} . \quad (9)$$ For each measurement function $\hat{h}_\sigma(\bullet)$ we determine the corresponding Jacobian function $$H_\sigma(\hat{x}(t), \hat{b}_t, \hat{e}_t)[i,j] \equiv \frac{\partial}{\partial \hat{x}[j]} \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{e}_t)[i] , \quad (10)$$ where $1 \le i \le m_\sigma$ and $1 \le j \le n$. Finally, we note the use of the standard (Kalman filter) $n \times n$ *error covariance matrix* $P(t)$ which maintains the covariance of the error in the estimated state. Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 6	Welch 1997

3.1.2 Tracking Algorithm

Given an initial state estimate $\hat{x}(0)$ and error covariance estimate $P(0)$, the SCAAT algorithm proceeds similarly to a conventional EKF, cycling through the following steps whenever a discrete measurement $\hat{z}_{\sigma,t}$ from some sensor (type σ) and source becomes available at time t:

a. Compute the time δt since the previous estimate.

b. Predict the state and error covariance.

$$\begin{aligned} \hat{x}^- &= A(\delta t)\hat{x}(t-\delta t) \\ P^- &= A(\delta t)P(t-\delta t)A^T(\delta t) + Q(\delta t) \end{aligned} \tag{11}$$

c. Predict the measurement and compute the corresponding Jacobian.

$$\begin{aligned} \hat{z} &= \hat{h}_\sigma(\hat{x}^-, \hat{b}_t, \vec{c}_t) \\ H &= H_\sigma(\hat{x}^-, \hat{b}_t, \hat{s}_t) \end{aligned} \tag{12}$$

d. Compute the *Kalman gain*.

$$K = P^- H^T (HP^- H^T + R_\sigma(t))^{-1} \tag{13}$$

e. Compute the *residual* between the actual sensor measurement $\hat{z}_{\sigma,t}$ and the predicted measurement from (12).

$$\vec{\Delta z} = \hat{z}_{\sigma,t} - \hat{z} \tag{14}$$

f. Correct the predicted tracker state estimate and error covariance from (11).

$$\begin{aligned} \hat{x}(t) &= \hat{x}^- + K\vec{\Delta z} \\ P(t) &= (I - KH)P^- \end{aligned} \tag{15}$$

Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 6	Welch 1997

g. Update the external orientation of equation (4) per the change indicated by the (ϕ, θ, ψ) elements of the state.

$$\Delta\hat{\alpha} = \text{quaternion}(\hat{x}[\phi], \hat{x}[\theta], \hat{x}[\psi])$$
$$\hat{\alpha} = \hat{\alpha} \otimes \Delta\hat{\alpha} \quad (16)$$

h. Zero the orientation elements of the state vector.

$$\hat{x}[\phi] = \hat{x}[\theta] = \hat{x}[\psi] = 0 \quad (17)$$

The equations (11)-(17) may seem computationally complex, however they can be performed quite efficiently. The computations can be optimized to eliminate operations on matrix and vector elements that are known to be zero. For example, the elements of the Jacobian H in (12) that correspond to the velocities in the state $\hat{x}(t)$ will always be zero. In addition, the matrix inverted in the computation of K in (13) is of rank m_σ (2×2 for our example in section 3.4) which is smaller for a SCAAT filter than for a corresponding conventional EKF implementation. Finally, the increased data rate allows the use of the *small angle approximations* $\sin(\theta) = \theta$ and $\cos(\theta) = 1$ in $\check{h}_\sigma(\bullet)$ and $H_\sigma(\bullet)$. The total *per estimate* computation time can therefore actually be less than that of a corresponding conventional implementation. (We are able to execute the SCAAT filter computations, with the autocalibration computations discussed in the next section, in approximately 100µs on a 200 MHz PC-compatible computer.)

Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 6	Welch 1997

3.1.3 Discussion

The key to the SCAAT method is the number of constraints provided by the measurement vector and measurement function in equation (8). For the 3D-tracking problem being solved, a unique solution requires $C = 6$ non-degenerate constraints to resolve six degrees of freedom. Because individual sensor measurements typically provide less than six constraints, conventional implementations usually construct a complete measurement vector

$$\hat{z}_t = \left[\hat{z}^T_{\sigma_1, t_1} \cdots \hat{z}^T_{\sigma_N, t_N} \right]^T$$

from some group of $N \geq C$ individual sensor measurements over time $t_1 \ldots t_N$, and *then* proceed to compute an estimate. Or a particular implementation may operate in a *moving-window* fashion, combining the most recent measurement with the $N-1$ previous measurements, possibly implementing a form of a finite-impulse-response filter. In any case, for such well-constrained systems complete observability is obtained at each step of the filter. Systems that collect measurements sequentially in this way inherently violate the simultaneity assumption, as well as increase the time δt between estimates.

In contrast, the SCAAT method blends individual measurements that each provide incomplete constraints into a complete state estimate. The EKF inherently provides the means for this blending, no matter how complete the information content of each individual measurement $\hat{z}_{\sigma, t}$. The EKF accomplishes this through the Kalman gain K which is computed in (13). The Kalman gain, which is used to adjust the state and the error covariance in (15), is optimal in the sense that it minimizes the error covariance if certain conditions are met. Note that the inversion in (13) forms a ratio that reflects the relative uncertainties of the state and the measurement. Note too that the ratio is affected by the use of the measurement function Jacobian H. Because the Jacobian reflects the rate of change of each measurement with respect to the current state, it indicates a direction in state space along which a measurement could *possibly* affect the state. Because the gain is recomputed at each step with the appropriate measurement function and associated Jacobian, it inherently reflects the amount and direction of information provided by the individual constraint.

Welch 1997 at 3.1.3.

Exhibit E-7

CLAIM 6	Welch 1997

3.2 Autocalibration

The method we use for autocalibration involves *augmenting* the *main tracker filter* presented in section 3.1 to effectively implement a distinct *device filter*, a Kalman filter, for each source or sensor to be calibrated. (We use the word "device" here to include for example scene landmarks which can be thought of as passive sources, and cameras which are indeed sensors.) In general, any constant device-related parameters used by a measurement function $\hat{h}_\sigma(\bullet)$ from (8) are candidates for this autocalibration method. We assume that the parameters to be estimated are contained in the device parameter vectors $\hat{b}_t$ and $\hat{c}_t$, and we also present the case where both the source and sensor are to be calibrated since omission of one or the other is trivial. We note the following new convention.

$\widehat{\alpha}$ = augmented matrix/vector (wide hat)

Welch 1997 at Section 3.2

3.2.1 Device Filters

For each device (source, sensor, landmark, etc.) we create a distinct device filter as follows. Let $\hat{\pi}$ represent the corresponding device parameter vector and n_π = length($\hat{\pi}$).

a. Allocate an $n_\pi \times 1$ state vector $\hat{x}_\pi$ for the device, initialize with the best *a priori* device parameter estimates, e.g. from design.

b. Allocate an $n_\pi \times n_\pi$ noise covariance matrix $Q_\pi(\delta t)$, initialize with the expected parameter variances.

c. Allocate an $n_\pi \times n_\pi$ error covariance matrix $P_\pi(t)$, initialize to indicate the level of confidence in the *a priori* device parameter estimates from (a) above.

Welch 1997 at Section 3.2.1.

Exhibit E-7

CLAIM 6	Welch 1997

3.2.2 Revised Tracking Algorithm

The algorithm for tracking with concurrent autocalibration is the same as that presented in section 3.1, with the following exceptions. After step (a) in the original algorithm, we form augmented versions of the state vector

$$\hat{\mathbf{x}}(t-\delta t) = \left[\hat{\mathbf{x}}^T(t-\delta t)\ \hat{\mathbf{x}}_{b,t}^T(t-\delta t)\ \hat{\mathbf{x}}_{c,t}^T(t-\delta t)\right]^T, \quad (18)$$

the error covariance matrix

$$\hat{\mathbf{P}}(t-\delta t) = \begin{bmatrix} \mathbf{P}(t-\delta t) & 0 & 0 \\ 0 & \mathbf{P}_{b,t}(t-\delta t) & 0 \\ 0 & 0 & \mathbf{P}_{c,t}(t-\delta t) \end{bmatrix}, \quad (19)$$

the state transition matrix

$$\hat{\mathbf{A}}(\delta t) = \begin{bmatrix} \mathbf{A}(\delta t) & 0 & 0 \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I} \end{bmatrix}, \quad (20)$$

and the process noise matrix

$$\hat{\mathbf{Q}}(\delta t) = \begin{bmatrix} \mathbf{Q}(\delta t) & 0 & 0 \\ 0 & \mathbf{Q}_{b,t}(\delta t) & 0 \\ 0 & 0 & \mathbf{Q}_{c,t}(\delta t) \end{bmatrix}. \quad (21)$$

Welch 1997 at Section 3.2.2.

Exhibit E-7

CLAIM 6	Welch 1997

We then follow steps (b)-(h) from the original algorithm, making the appropriate substitutions of (18)-(21), and noting that the measurement and Jacobian functions used in step (c) have become $\hat{h}_\sigma(\hat{\vec{x}}(t))$ and $H_\sigma(\hat{\vec{x}}(t))$ because the estimates of parameters $\hat{b}_t$ and $\hat{\delta}_t$ ($\hat{x}_{b,t}$ and $\hat{x}_{c,t}$) are now contained in the augmented state vector $\hat{\vec{x}}$ per (18). After step (h) we finish by extracting and saving the device filter portions of the augmented state vector and error covariance matrix

$$\begin{aligned} \hat{x}_{b,t}(t) &= \hat{\vec{x}}(t)[i..j] \\ P_{b,t}(t) &= \hat{P}(t)[i..j, i..j] \\ \hat{x}_{c,t}(t) &= \hat{\vec{x}}(t)[k..l] \\ P_{c,t}(t) &= \hat{P}(t)[k..l, k..l] \end{aligned} \tag{22}$$

where

$$\begin{aligned} i &= n+1 \\ j &= n+n_b \\ k &= n+n_b+1 \\ l &= n+n_b+n_c \end{aligned}$$

and n, n_b, and n_c are the dimensions of the state vectors for the main tracker filter, the source filter, and the sensor filter (respectively). We leave the main tracker filter state vector and error covariance matrix in their augmented counterparts, while we swap the device filter components in and out with each estimate. The result is that individual device filters are updated less frequently than the main tracker filter. The more a device is used, the more it is calibrated. If a device is never used, it is never calibrated.

Welch 1997 at Section 3.2.2.

Exhibit E-7

CLAIM 6	Welch 1997

With respect to added time complexity, the computations can again be optimized to eliminate operations on matrix and vector elements that are known to be zero: those places mentioned in section 3.1, and see (19)-(21). Also note that the size of and thus time for the matrix inversion in (13) has not changed. With respect to added space complexity, the autocalibration method requires storing a separate state vector and covariance matrix for each device—a fixed amount of (generally small) space per device. For example, consider autocalibrating the beacon (LED) positions for an optical tracking system with 3,000 beacons. For each beacon one would need 3 words for the beacon state (its 3D position), $3 \times 3 = 9$ words for the noise covariance matrix, and $3 \times 3 = 9$ words for the error covariance matrix. Assuming 8 bytes per word, this is only $3{,}000 \times 8 \times (3 + 9 + 9) = 504{,}000$ bytes.

Welch 1997 at Section 3.2.2.

3.2.3 Discussion

The ability to simultaneously estimate two dependent sets of unknowns (the target and device states) is made possible by several factors. First, the dynamics of the two sets are very different as would be reflected in the process noise matrices. We assume the target is undergoing some random (constant) acceleration, reflected in the noise parameter η of $Q(\delta t)$ in (6). Conversely, we assume the device parameters are constant, and so the elements of $Q_\pi(\delta t)$ for a source or sensor simply reflect any allowed variances in the corresponding parameters: usually zero or extremely small. In addition, while the target is expected to be moving, the filter expects the motion between any two estimations to closely correspond to the velocity estimates in the state (3). If the tracker estimate rate is high enough, poorly estimated device parameters will result in what appears to be almost instantaneous target motion. The increased rate of the SCAAT method allows such motion to be recognized as unlikely, and attributed to poorly estimated device parameters.

Welch 1997 at Section 3.2.3

3.3 Stability

Because the SCAAT method uses individual measurements with insufficient information, one might be concerned about the potential for instability or divergence. A linear system is said to be stable if its response to

CLAIM 6	Welch 1997
	any input tends to a finite steady value after the input is removed [24]. For the Kalman filter in general this is certainly not a new concern, and there are standard requirements and corresponding tests that ensure or detect stability (see [18], p. 132): a. The filter must be uniformly completely observable, b. the dynamic and measurement noise matrices in equations (6) and (9) must be bounded from above and below, and c. the dynamic behavior represented by in equation (2) must be bounded from above. As it turns out, these conditions and their standard tests are equally applicable to a SCAAT implementation. For the SCAAT method the conditions mean that the user dynamics between estimates must be bounded, the measurement noise must be bounded, one must incorporate a sufficient set of non-degenerate constraints over time. In particular, the constraints must be incorporated in less than 1/2 the time of the user motion time-constant in order to avoid tracking an alias of the true motion. In general these conditions are easily met for systems and circumstances that would otherwise be stable with a multiple-constraint implementation. A complete stability analysis is beyond the scope of this paper, and is presented in [47]. Welch 1997 at Section 3.3. [47] Greg Welch, 1996. "SCAAT: Incremental Tracking with Incomplete Information," University of North Carolina at Chapel Hill, doctoral dissertation, TR 96-051. Welch 1997 at References. *See* Disclosures with respect to Claim 1, *supra*; *see also* Defendants' Invalidity Contentions for further discussion.
[6.a] enumerating sensing elements available to a tracking system that includes an estimation subsystem that estimates a position or orientation of an object; and	At least under Plaintiffs' apparent infringement theory, Welch 1997 discloses, either expressly or inherently, enumerating sensing elements available to a tracking system that includes an estimation subsystem that estimates a position or orientation of an object. In the alternative, this element would be obvious over Welch 1997 in light of the other references disclosed in Defendants' Invalidity Contentions and/or the knowledge of one of ordinary skill in the art. *See, e.g.*: We present a promising new mathematical method for tracking a user's pose (position and orientation) for interactive computer graphics. The method, which is applicable to a wide variety of both commercial and

CLAIM 6	Welch 1997
	experimental systems, improves accuracy by properly assimilating sequential observations, filtering sensor measurements, and by concurrently autocalibrating source and sensor devices. It facilitates user motion prediction, multisensor data fusion, and higher report rates with lower latency than previous methods. Tracking systems determine the user's pose by measuring signals from low-level hardware sensors. For reasons of physics and economics, most systems make multiple sequential measurements which are then combined to produce a single tracker report. For example, commercial magnetic trackers using the SPASYN (Space Synchro) system sequentially measure three magnetic vectors and then combine them mathematically to produce a report of the sensor pose. Our new approach produces tracker reports as each new low-level sensor measurement is made rather than waiting to form a complete collection of observations. Because single observations under-constrain the mathematical solution, we refer to our approach as single-constraint-at-a-time or SCAAT tracking. The key is that the single observations provide some information about the user's state, and thus can be used to incrementally improve a previous estimate. We recursively apply this principle, incorporating new sensor data as soon as it is measured. With this approach we are able to generate estimates more frequently, with less latency, and with improved accuracy. We present results from both an actual implementation, and from extensive simulations. Welch 1997 at Abstract.

Exhibit E-7

CLAIM 6	Welch 1997

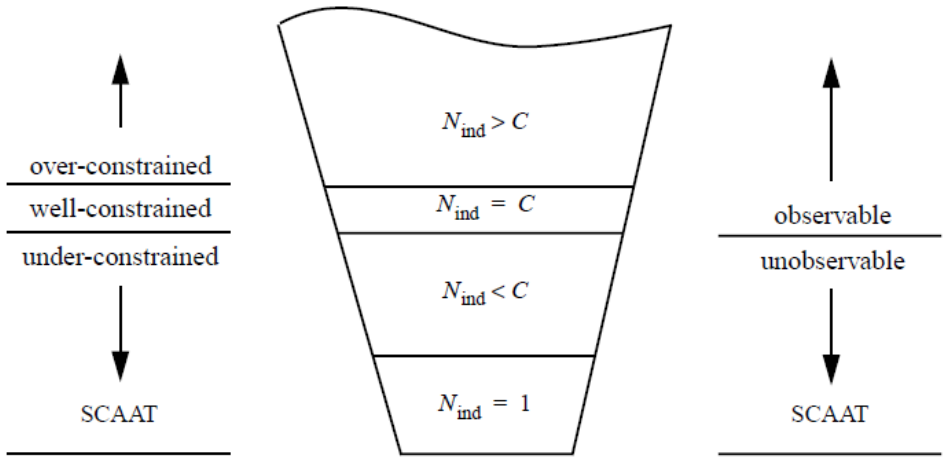

Figure 1: SCAAT and constraints on a system of simultaneous equations. C is the minimal number of independent simultaneous constraints necessary to uniquely determine a solution, N is the number of given constraints, and N_{ind} is the number of *independent* constraints that can be formed from the N. (For most systems of interest $C > 1$). The conventional approach is to ensure $N \geq N_{ind}$ and $N_{ind} \geq C$, i.e. to use enough measurements to well-constrain or even over-constrain the estimate. The SCAAT approach is to employ the smallest number of constraints available at any one time, generally $N = N_{ind} = 1$ constraint. From this viewpoint, each SCAAT estimate is severely under-constrained.

Welch 1997 at Figure 1.

Exhibit E-7

CLAIM 6	Welch 1997

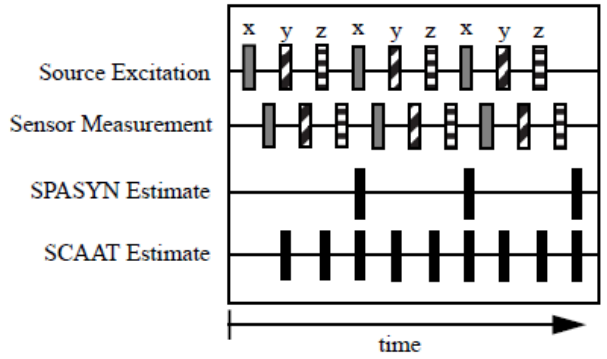

Figure 2: A timing diagram comparing the SPASYN (Polhemus Navigation Systems) magnetic position and orientation tracking system with a hypothetical SCAAT implementation.

Welch 1997 at Figure 2.

The SCAAT method employs a Kalman filter (KF) in an unusual fashion. The Kalman filter is a mathematical procedure that provides an efficient computational (recursive) method for the least-squares estimation of a linear system. It does so in a predictor-corrector fashion, predicting short-term (since the last estimate) changes in the state using a dynamic model, and then correcting them with a measurement and a corresponding measurement model. The extended Kalman filter (EKF) is a variation of the Kalman filter that supports estimation of nonlinear systems, e.g. 3D position and orientation tracking systems. A basic introduction to the Kalman filter can be found in Chapter 1 of [31], while a more complete introductory discussion can be found in [40], which also contains some interesting historical narrative. More extensive references can be found in [7,18,24,28,31,46].
Welch 1997 at Section 3.

The Kalman filter has been employed previously for virtual environment tracking estimation and prediction. For example see [2,5,12,14,42], and most recently [32]. In each of these cases however the filter was applied directly and only to the 6D pose estimates delivered by the off-the-shelf tracker. The SCAAT approach could be applied to either a hybrid system using off-the-shelf and/or custom trackers, or it could be employed by tracker developers to improve the existing systems for the end-user graphics community.
Welch 1997 at Section 3.

Exhibit E-7

CLAIM 6	Welch 1997
	In this section we describe the method in a manner that does not imply a specific tracking system. (In section 3.4 we present experimental results of a specific implementation, a SCAAT wide-area optoelectronic tracking system.) In section 3.1 we describe the method for tracking, and in section 3.2 we describe one possible method for concurrent autocalibration. Welch 1997 at Section 3. Throughout we use the following conventions. x = scalar (lower case) $\vec{x}$ = general vector (lower case, arrow) indexed as $\vec{x}[r]$ $\hat{x}$ = filter estimate vector (lower case, hat) A = matrix (capital letters) indexed as $A[r, c]$ A^{-1} = matrix inverse I = the identity matrix β^{-} = matrix/vector *prediction* (super minus) β^{T} = matrix/vector transpose (super T) α_i = matrix/vector/scalar identifier (subscript) $E\{\bullet\}$ = mathematical expectation Welch 1997 at Section 3.

Exhibit E-7

CLAIM 6	Welch 1997
	3.1 Tracking **3.1.1 Main Tracker Filter** The use of a Kalman filter requires a mathematical (state-space) model for the dynamics of the process to be estimated, the target motion in this case. While several possible dynamic models and associated state configurations are possible, we have found a simple *position-velocity* model to suffice for the dynamics of our applications. In fact we use this same form of model, with different parameters, for all six of the position and orientation components $(x, y, z, \phi, \theta, \psi)$. Discussion of some other potential models and the associated trade-offs can be found in [7] pp. 415-420. Because our implementation is discrete with inter sample time δt we model the target's dynamic motion with the following linear difference equation: $$\hat{x}(t+\delta t) = A(\delta t)\hat{x}(t) + \hat{w}(\delta t). \quad (2)$$ Welch 1997 at Section 3.1.

Exhibit E-7

CLAIM 6	Welch 1997
	In the standard model corresponding to equation (2), the n dimensional Kalman filter *state vector* $\hat{x}(t)$ would completely describe the target position and orientation at any time t. In practice we use a method similar to [2,6] and maintain the complete target orientation externally to the Kalman filter in order to avoid the nonlinearities associated with orientation computations. In the internal state vector $\hat{x}(t)$ we maintain the target position as the Cartesian coordinates (x, y, z), and the *incremental* orientation as small rotations (ϕ, θ, ψ) about the (x, y, z) axis. Externally we maintain the target orientation as the *external quaternion* $\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z))$. (See [9] for discussion of quaternions.) At each filter update step, the incremental orientations (ϕ, θ, ψ) are factored into the external quaternion $\hat{\alpha}$, and then zeroed as shown below. Thus the incremental orientations are linearized for the EKF, centered about zero. We maintain the derivatives of the target position and orientation internally, in the state vector $\hat{x}(t)$. We maintain the angular velocities internally because the angular velocities behave like orthogonal vectors and do not exhibit the nonlinearities of the angles themselves. The target state is then represented by the $n = 12$ element internal state vector $$\hat{x} = \left[x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \phi\ \theta\ \psi\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}\right]^T \quad (3)$$ and the four-element external orientation quaternion $$\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z)), \quad (4)$$ where the time designations have been omitted for clarity. Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 6	Welch 1997

The $n \times n$ *state transition matrix* $A(\delta t)$ in (2) projects the state forward from time t to time $t + \delta t$. For our linear model, the matrix implements the relationships

$$\begin{aligned} x(t+\delta t) &= x(t) + \dot{x}(t)\delta t \\ \dot{x}(t+\delta t) &= \dot{x}(t) \end{aligned} \qquad (5)$$

and likewise for the remaining elements of (3).

The $n \times 1$ *process noise vector* $\vec{w}(\delta t)$ in (2) is a normally-distributed zero-mean sequence that represents the uncertainty in the target state over any time interval δt. The corresponding $n \times n$ *process noise covariance matrix* is given by

$$E\{\vec{w}(\delta t)\vec{w}^T(\delta t + \varepsilon)\} = \begin{cases} Q(\delta t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \qquad (6)$$

Because our implementation is discrete with inter sample time δt, we can use the transfer function method illustrated by [7] pp. 221-222 to compute a *sampled* process noise covariance matrix. (Because the associated random processes are presumed to be time stationary, we present the process noise covariance matrix as a function of the inter-sample duration δt only.) The non-zero elements of $Q(\delta t)$ are given by

$$\begin{aligned} Q(\delta t)[i,i] &= \vec{\eta}[i]\frac{(\delta t)^3}{3} \\ Q(\delta t)[i,j] = Q(\delta t)[j,i] &= \vec{\eta}[i]\frac{(\delta t)^2}{2} \\ Q(\delta t)[j,j] &= \vec{\eta}[i](\delta t) \end{aligned} \qquad (7)$$

for each pair

$$(i,j) \in \{(x,\dot{x}), (y,\dot{y}), (z,\dot{z}), (\phi,\dot{\phi}), (\theta,\dot{\theta}), (\psi,\dot{\psi})\}.$$

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 6	Welch 1997

The $\hat{\eta}[i]$ in (7) are the *correlation kernels* of the (assumed constant) noise sources presumed to be driving the dynamic model. We determined a set of values using Powell's method, and then used these in both simulation and our real implementation. The values can be "tuned" for different dynamics, though we have found that the tracker works well over a broad range of values.

The use of a Kalman filter requires not only a dynamic model as described above, but also a *measurement model* for each available type of measurement. The measurement model is used to predict the ideal noise-free response of each sensor and source pair, given the filter's current estimate of the target state as in equations (3) and (4).

It is the nature of the measurement models and indeed the actual sensor measurements that distinguishes a SCAAT Kalman filter from a well-constrained one.

For each sensor *type* σ we define the $m_\sigma \times 1$ *measurement vector* $\hat{z}_\sigma(t)$ and corresponding *measurement function* $\hat{h}_\sigma(\bullet)$ such that

$$\hat{z}_{\sigma,t} = \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t) + \hat{v}_\sigma(t) . \qquad (8)$$

Note that in the "purest" SCAAT implementation $m_\sigma = 1$ and the measurements are incorporated as single scalar values. However if it is not possible or necessary to isolate the measurements, e.g. to perform autocalibration, then multi-dimensional measurements can be incorporated also. Guidelines presented in [47] lead to the following heuristic for choosing the SCAAT Kalman filter measurement elements (constraints):

During each SCAAT Kalman filter measurement update one should observe a single sensor and source pair only.

For example, to incorporate magnetic tracker data as an end-user, $m_\sigma = 7$ for the three position and four orientation (quaternion)

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 6	Welch 1997

elements, while if the manufacturer were to use the SCAAT implementation, $m_\sigma = 3$ for each 3-axis electromagnetic response to a single excitation. For an image-based landmark tracker such as [41] the measurement function would, given estimates of the camera pose and a single landmark location, transform the landmark into camera space and then project it onto the camera image plane. In this case $m_\sigma = 2$ for the 2D image coordinates of the landmark.

The $m_\sigma \times 1$ *measurement noise vector* $\hat{\nu}_\sigma(t)$ in (8) is a normally-distributed zero-mean sequence that represents any random error (e.g. electrical noise) in the measurement. This parameter can be determined from component design specifications, and (or) confirmed by off-line measurement. For our simulations we did both. The corresponding $m_\sigma \times m_\sigma$ *measurement noise covariance matrix* is given by

$$E\{\hat{\nu}_\sigma(t)\hat{\nu}_\sigma^T(t+\varepsilon)\} = \begin{cases} R_\sigma(t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \qquad (9)$$

For each measurement function $\hat{h}_\sigma(\bullet)$ we determine the corresponding Jacobian function

$$H_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t)[i,j] \equiv \frac{\partial}{\partial \hat{x}[j]} \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t)[i], \qquad (10)$$

where $1 \le i \le m_\sigma$ and $1 \le j \le n$. Finally, we note the use of the standard (Kalman filter) $n \times n$ *error covariance matrix* $P(t)$ which maintains the covariance of the error in the estimated state.

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 6	Welch 1997

3.1.2 Tracking Algorithm

Given an initial state estimate $\hat{x}(0)$ and error covariance estimate $P(0)$, the SCAAT algorithm proceeds similarly to a conventional EKF, cycling through the following steps whenever a discrete measurement $\vec{z}_{\sigma,t}$ from some sensor (type σ) and source becomes available at time t:

a. Compute the time δt since the previous estimate.

b. Predict the state and error covariance.

$$\begin{aligned} \hat{x}^- &= A(\delta t)\hat{x}(t-\delta t) \\ P^- &= A(\delta t)P(t-\delta t)A^T(\delta t) + Q(\delta t) \end{aligned} \tag{11}$$

c. Predict the measurement and compute the corresponding Jacobian.

$$\begin{aligned} \hat{z} &= \vec{h}_\sigma(\hat{x}^-, \vec{b}_t, \vec{c}_t) \\ H &= H_\sigma(\hat{x}^-, \vec{b}_t, \vec{e}_t) \end{aligned} \tag{12}$$

d. Compute the *Kalman gain*.

$$K = P^-H^T(HP^-H^T + R_\sigma(t))^{-1} \tag{13}$$

e. Compute the *residual* between the actual sensor measurement $\vec{z}_{\sigma,t}$ and the predicted measurement from (12).

$$\vec{\Delta z} = \vec{z}_{\sigma,t} - \hat{z} \tag{14}$$

f. Correct the predicted tracker state estimate and error covariance from (11).

$$\begin{aligned} \hat{x}(t) &= \hat{x}^- + K\vec{\Delta z} \\ P(t) &= (I-KH)P^- \end{aligned} \tag{15}$$

Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 6	Welch 1997
	g. Update the external orientation of equation (4) per the change indicated by the (ϕ, θ, ψ) elements of the state.* $\Delta\hat{\alpha} = \text{quaternion}(\hat{x}[\phi], \hat{x}[\theta], \hat{x}[\psi])$ $\hat{\alpha} = \hat{\alpha} \otimes \Delta\hat{\alpha}$ (16) h. Zero the orientation elements of the state vector. $\hat{x}[\phi] = \hat{x}[\theta] = \hat{x}[\psi] = 0$ (17) The equations (11)-(17) may seem computationally complex, however they can be performed quite efficiently. The computations can be optimized to eliminate operations on matrix and vector elements that are known to be zero. For example, the elements of the Jacobian H in (12) that correspond to the velocities in the state $\hat{x}(t)$ will always be zero. In addition, the matrix inverted in the computation of K in (13) is of rank m_σ (2×2 for our example in section 3.4) which is smaller for a SCAAT filter than for a corresponding conventional EKF implementation. Finally, the increased data rate allows the use of the *small angle approximations* $\sin(\theta) = \theta$ and $\cos(\theta) = 1$ in $\check{h}_\sigma(\bullet)$ and $H_\sigma(\bullet)$. The total *per estimate* computation time can therefore actually be less than that of a corresponding conventional implementation. (We are able to execute the SCAAT filter computations, with the autocalibration computations discussed in the next section, in approximately 100µs on a 200 MHz PC-compatible computer.) Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 6	Welch 1997

3.1.3 Discussion

The key to the SCAAT method is the number of constraints provided by the measurement vector and measurement function in equation (8). For the 3D-tracking problem being solved, a unique solution requires $C = 6$ non-degenerate constraints to resolve six degrees of freedom. Because individual sensor measurements typically provide less than six constraints, conventional implementations usually construct a complete measurement vector

$$\hat{z}_t = \left[\hat{z}^T_{\sigma_1, t_1} \cdots \hat{z}^T_{\sigma_N, t_N}\right]^T$$

from some group of $N \geq C$ individual sensor measurements over time $t_1 \ldots t_N$, and *then* proceed to compute an estimate. Or a particular implementation may operate in a *moving-window* fashion, combining the most recent measurement with the $N-1$ previous measurements, possibly implementing a form of a finite-impulse-response filter. In any case, for such well-constrained systems complete observability is obtained at each step of the filter. Systems that collect measurements sequentially in this way inherently violate the simultaneity assumption, as well as increase the time δt between estimates.

In contrast, the SCAAT method blends individual measurements that each provide incomplete constraints into a complete state estimate. The EKF inherently provides the means for this blending, no matter how complete the information content of each individual measurement $\hat{z}_{\sigma, t}$. The EKF accomplishes this through the Kalman gain K which is computed in (13). The Kalman gain, which is used to adjust the state and the error covariance in (15), is optimal in the sense that it minimizes the error covariance if certain conditions are met. Note that the inversion in (13) forms a ratio that reflects the relative uncertainties of the state and the measurement. Note too that the ratio is affected by the use of the measurement function Jacobian H. Because the Jacobian reflects the rate of change of each measurement with respect to the current state, it indicates a direction in state space along which a measurement could *possibly* affect the state. Because the gain is recomputed at each step with the appropriate measurement function and associated Jacobian, it inherently reflects the amount and direction of information provided by the individual constraint.

Welch 1997 at 3.1.3.

Exhibit E-7

CLAIM 6	Welch 1997

3.2 Autocalibration

The method we use for autocalibration involves *augmenting* the *main tracker filter* presented in section 3.1 to effectively implement a distinct *device filter*, a Kalman filter, for each source or sensor to be calibrated. (We use the word "device" here to include for example scene landmarks which can be thought of as passive sources, and cameras which are indeed sensors.) In general, any constant device-related parameters used by a measurement function $\hat{h}_\sigma(\bullet)$ from (8) are candidates for this autocalibration method. We assume that the parameters to be estimated are contained in the device parameter vectors $\hat{b}_t$ and $\hat{c}_t$, and we also present the case where both the source and sensor are to be calibrated since omission of one or the other is trivial. We note the following new convention.

$\widehat{\alpha}$ = augmented matrix/vector (wide hat)

Welch 1997 at Section 3.2.

3.2.1 Device Filters

For each device (source, sensor, landmark, etc.) we create a distinct device filter as follows. Let $\hat{\pi}$ represent the corresponding device parameter vector and $n_\pi = \text{length}(\hat{\pi})$.

a. Allocate an $n_\pi \times 1$ state vector $\hat{x}_\pi$ for the device, initialize with the best *a priori* device parameter estimates, e.g. from design.

b. Allocate an $n_\pi \times n_\pi$ noise covariance matrix $Q_\pi(\delta t)$, initialize with the expected parameter variances.

c. Allocate an $n_\pi \times n_\pi$ error covariance matrix $P_\pi(t)$, initialize to indicate the level of confidence in the *a priori* device parameter estimates from (a) above.

Welch 1997 at Section 3.2.1.

Exhibit E-7

CLAIM 6	Welch 1997

3.2.2 Revised Tracking Algorithm

The algorithm for tracking with concurrent autocalibration is the same as that presented in section 3.1, with the following exceptions. After step (a) in the original algorithm, we form augmented versions of the state vector

$$\widehat{x}(t-\delta t) = \left[\hat{x}^T(t-\delta t)\ \hat{x}_{b,t}^T(t-\delta t)\ \hat{x}_{c,t}^T(t-\delta t)\right]^T, \quad (18)$$

the error covariance matrix

$$\widehat{P}(t-\delta t) = \begin{bmatrix} P(t-\delta t) & 0 & 0 \\ 0 & P_{b,t}(t-\delta t) & 0 \\ 0 & 0 & P_{c,t}(t-\delta t) \end{bmatrix}, \quad (19)$$

the state transition matrix

$$\widehat{A}(\delta t) = \begin{bmatrix} A(\delta t) & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}, \quad (20)$$

and the process noise matrix

$$\widehat{Q}(\delta t) = \begin{bmatrix} Q(\delta t) & 0 & 0 \\ 0 & Q_{b,t}(\delta t) & 0 \\ 0 & 0 & Q_{c,t}(\delta t) \end{bmatrix}. \quad (21)$$

Welch 1997 at Section 3.2.2.

Exhibit E-7

CLAIM 6	Welch 1997
	We then follow steps (b)-(h) from the original algorithm, making the appropriate substitutions of (18)-(21), and noting that the measurement and Jacobian functions used in step (c) have become $\hat{h}_\sigma(\widehat{x}(t))$ and $H_\sigma(\widehat{x}(t))$ because the estimates of parameters $\hat{b}_t$ and $\hat{c}_t$ ($\hat{x}_{b,t}$ and $\hat{x}_{c,t}$) are now contained in the augmented state vector $\widehat{x}$ per (18). After step (h) we finish by extracting and saving the device filter portions of the augmented state vector and error covariance matrix $\hat{x}_{b,t}(t) = \widehat{x}(t)[i\ldots j]$ $P_{b,t}(t) = \widehat{P}(t)[i\ldots j, i\ldots j]$ (22) $\hat{x}_{c,t}(t) = \widehat{x}(t)[k\ldots l]$ $P_{c,t}(t) = \widehat{P}(t)[k\ldots l, k\ldots l]$ where $i = n+1$ $j = n+n_b$ $k = n+n_b+1$ $l = n+n_b+n_c$ Welch 1997 at Section 3.2.2.

Exhibit E-7

CLAIM 6	Welch 1997

and n, n_b, and n_c are the dimensions of the state vectors for the main tracker filter, the source filter, and the sensor filter (respectively). We leave the main tracker filter state vector and error covariance matrix in their augmented counterparts, while we swap the device filter components in and out with each estimate. The result is that individual device filters are updated less frequently than the main tracker filter. The more a device is used, the more it is calibrated. If a device is never used, it is never calibrated.

With respect to added time complexity, the computations can again be optimized to eliminate operations on matrix and vector elements that are known to be zero: those places mentioned in section 3.1, and see (19)-(21). Also note that the size of and thus time for the matrix inversion in (13) has not changed. With respect to added space complexity, the autocalibration method requires storing a separate state vector and covariance matrix for each device—a fixed amount of (generally small) space per device. For example, consider autocalibrating the beacon (LED) positions for an optical tracking system with 3,000 beacons. For each beacon one would need 3 words for the beacon state (its 3D position), $3 \times 3 = 9$ words for the noise covariance matrix, and $3 \times 3 = 9$ words for the error covariance matrix. Assuming 8 bytes per word, this is only $3{,}000 \times 8 \times (3 + 9 + 9) = 504{,}000$ bytes.

Welch 1997 at Section 3.2.2.

3.2.3 Discussion

The ability to simultaneously estimate two dependent sets of unknowns (the target and device states) is made possible by several factors. First, the dynamics of the two sets are very different as would be reflected in the process noise matrices. We assume the target is undergoing some random (constant) acceleration, reflected in the noise parameter η of $Q(\delta t)$ in (6). Conversely, we assume the device parameters are constant, and so the elements of $Q_\pi(\delta t)$ for a source or sensor simply reflect any allowed variances in the corresponding parameters: usually zero or extremely small. In addition, while the target is expected to be moving, the filter expects the motion between any two estimations to closely correspond to the velocity estimates in the state (3). If the tracker estimate rate is high enough, poorly estimated device parameters will result in what appears to be almost instantaneous target motion. The increased rate of the SCAAT method allows such motion to be recognized as unlikely, and attributed to poorly estimated device parameters.

Welch 1997 at Section 3.2.3.

Exhibit E-7

CLAIM 6	Welch 1997
	3.3 Stability Because the SCAAT method uses individual measurements with insufficient information, one might be concerned about the potential for instability or divergence. A linear system is said to be stable if its response to any input tends to a finite steady value after the input is removed [24]. For the Kalman filter in general this is certainly not a new concern, and there are standard requirements and corresponding tests that ensure or detect stability (see [18], p. 132): a. The filter must be uniformly completely observable, b. the dynamic and measurement noise matrices in equations (6) and (9) must be bounded from above and below, and c. the dynamic behavior represented by in equation (2) must be bounded from above. As it turns out, these conditions and their standard tests are equally applicable to a SCAAT implementation. For the SCAAT method the conditions mean that the user dynamics between estimates must be bounded, the measurement noise must be bounded, one must incorporate a sufficient set of non-degenerate constraints over time. In particular, the constraints must be incorporated in less than 1/2 the time of the user motion time-constant in order to avoid tracking an alias of the true motion. In general these conditions are easily met for systems and circumstances that would otherwise be stable with a multiple-constraint implementation. A complete stability analysis is beyond the scope of this paper, and is presented in [47]. Welch 1997 at Section 3.3. [47] Greg Welch, 1996. “SCAAT: Incremental Tracking with Incomplete Information,” University of North Carolina at Chapel Hill, doctoral dissertation, TR 96-051. Welch 1997 at References.

Exhibit E-7

CLAIM 6	Welch 1997
	 Figure 3: The HiBall is shown here with the internal circuitry exposed and the lenses removed. The sensors, which can be seen through the lens openings, are mounted on PC boards that fold-up into the HiBall upon assembly. The mechanical pencil at the bottom conveys an indication of the relative size of the unit. Welch 1997 at Figure 3.

CLAIM 6	Welch 1997
	4.1 Tracker Filter The 12 element state vector $\hat{x}(t)$ for the main tracker filter contained the elements shown in (3). Each of the 3000 beacon filters was allocated a 3 element state vector $$\hat{x}_b = \left[x_b\ y_b\ z_b\right]^T$$ where (x_b, y_b, z_b) represents the beacon's estimated position in cartesian (world) coordinates. The 12×12 state transition matrix for the main tracker filter was formed as discussed section 3.1, and for each beacon filter it was the 3×3 identity matrix. The 12×12 process noise matrix for the main tracker was computed using (7), using elements of $\hat{\eta}$ that were determined off-line using Powell's method and a variety of real motion data. For each beacon filter we used an identical noise covariance matrix $$Q_b(\delta t)[i,j] = \begin{cases} \eta_b & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$ for $1 \le i, j \le 3$, with beacon position variance η_b also determined off-line. (See [47] for the complete details.) At each estimate step, the *augmented* 15 element state vector, 15×15 process noise matrix, 15×15 state transition matrix, and 15×15 error covariance matrix all resembled (18)-(21) (without the camera parameter components). The measurement noise model was distance dependent (beacon light falls-off with distance) so $R_\sigma(t)$ from (9) was computed prior to step (d), by using a beacon distance estimate (obtained from the user and beacon positions in the predicted state $\hat{x}^-$) to project a distance-dependent electrical variance onto the camera. Welch 1997 at Section 4.1. *See* Disclosures with respect to Claim 1, *supra*; *see also* Defendants' Invalidity Contentions for further discussion.
[6.b] providing parameters specific to the enumerated sensing elements to the tracking system to enable the estimation subsystem to be configured based on the parameters specific to	At least under Plaintiffs' apparent infringement theory, Welch 1997 discloses, either expressly or inherently, providing parameters specific to the enumerated sensing elements to the tracking system to enable the estimation subsystem to be configured based on the parameters specific to the enumerated sensing elements to enable the estimation subsystem to estimate the position or orientation of the object. In the alternative, this element would be obvious over Welch 1997 in light of the other references disclosed in Defendants' Invalidity Contentions and/or the knowledge of one of ordinary skill in the art. *See, e.g.*:

CLAIM 6	Welch 1997
the enumerated sensing elements to enable the estimation subsystem to estimate the position or orientation of the object.	2.2 Device Isolation & Autocalibration **Knowledge about source and sensor imperfections can be used to improve the accuracy of tracking systems. While intrinsic sensor parameters can often be determined off-line, e.g. by the manufacturer, this is generally not the case for extrinsic parameters.** For example it can be difficult to determine the exact geometric relationship between the various sensors of a hybrid system. Consider that the coordinate system of a magnetic sensor is located at some unknown location inside the sensor unit. Similarly the precise geometric relationship between visible landmarks used in a vision-based system is often difficult to determine. Even worse, landmark positions can change over time as, for example, a patient's skin deforms with pressure from an ultrasound probe. Welch 1997 at 2.2. In general, goals such as flexibility, ease of use, and lower cost, make the notion of self-calibration or autocalibration attractive. The general idea for autocalibration is not new. See for example [19,45]. **However, because the SCAAT method isolates the measurements provided by each sensor or modality, the method provides a new and elegant means to autocalibrate concurrently while tracking. Because the SCAAT method isolates the individual measurements, or measurement dimensions, individual source and sensor imperfections are more easily identified and dealt with.** Furthermore, because the simultaneity assumption is avoided, the motion restrictions discussed in section 2.1 would be removed, and autocalibration could be performed while concurrently tracking a target. Welch 1997 at 2.2. The isolation enforced by the SCAAT approach can improve results even if the constraints are obtained simultaneously through multidimensional measurements. An intuitive explanation is that if the elements (dimensions) are corrupted by independent noise, then incorporating the elements independently can offer improved filtering over a batch or ensemble estimation scheme. Welch 1997 at 2.2.

Exhibit E-7

CLAIM 6	Welch 1997

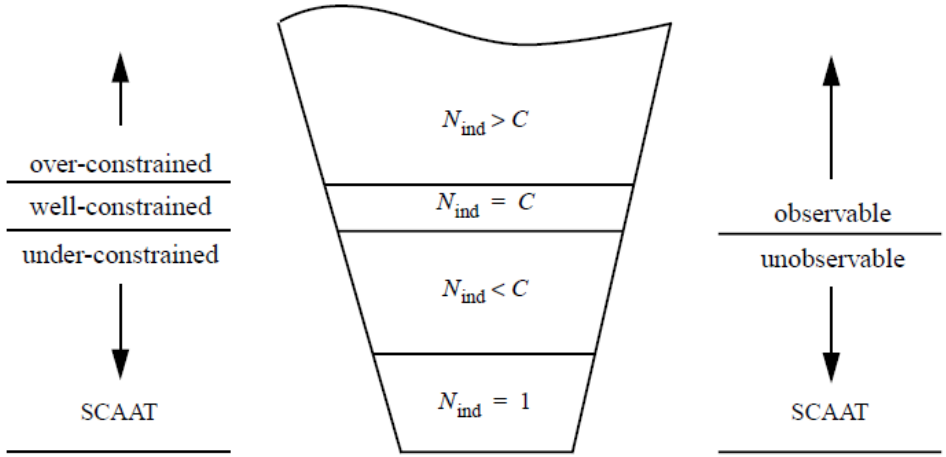

Figure 1: SCAAT and constraints on a system of simultaneous equations. C is the minimal number of independent simultaneous constraints necessary to uniquely determine a solution, N is the number of given constraints, and N_{ind} is the number of *independent* constraints that can be formed from the N. (For most systems of interest $C > 1$). The conventional approach is to ensure $N \geq N_{ind}$ and $N_{ind} \geq C$, i.e. to use enough measurements to well-constrain or even over-constrain the estimate. The SCAAT approach is to employ the smallest number of constraints available at any one time, generally $N = N_{ind} = 1$ constraint. From this viewpoint, each SCAAT estimate is severely under-constrained.

Welch 1997 at Figure 1.

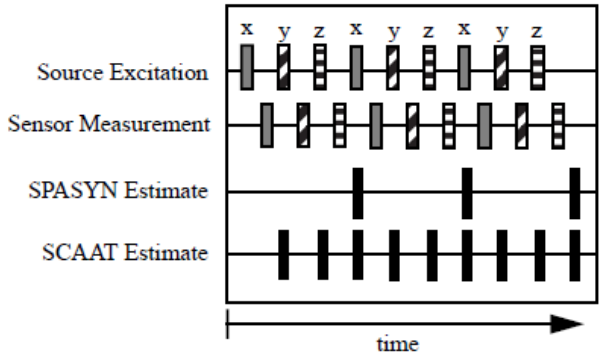

Figure 2: A timing diagram comparing the SPASYN (Polhemus Navigation Systems) magnetic position and orientation tracking system with a hypothetical SCAAT implementation.

Exhibit E-7

CLAIM 6	Welch 1997
	Welch 1997 at Figure 2. The SCAAT method employs a Kalman filter (KF) in an unusual fashion. The Kalman filter is a mathematical procedure that provides an efficient computational (recursive) method for the least-squares estimation of a linear system. It does so in a predictor-corrector fashion, predicting short-term (since the last estimate) changes in the state using a dynamic model, and then correcting them with a measurement and a corresponding measurement model. The extended Kalman filter (EKF) is a variation of the Kalman filter that supports estimation of nonlinear systems, e.g. 3D position and orientation tracking systems. A basic introduction to the Kalman filter can be found in Chapter 1 of [31], while a more complete introductory discussion can be found in [40], which also contains some interesting historical narrative. More extensive references can be found in [7,18,24,28,31,46]. Welch 1997 at Section 3. The Kalman filter has been employed previously for virtual environment tracking estimation and prediction. For example see [2,5,12,14,42], and most recently [32]. In each of these cases however the filter was applied directly and only to the 6D pose estimates delivered by the off-the-shelf tracker. The SCAAT approach could be applied to either a hybrid system using off-the-shelf and/or custom trackers, or it could be employed by tracker developers to improve the existing systems for the end-user graphics community. Welch 1997 at Section 3. In this section we describe the method in a manner that does not imply a specific tracking system. (In section 3.4 we present experimental results of a specific implementation, a SCAAT wide-area optoelectronic tracking system.) In section 3.1 we describe the method for tracking, and in section 3.2 we describe one possible method for concurrent autocalibration. Welch 1997 at Section 3.

Exhibit E-7

CLAIM 6	Welch 1997

Throughout we use the following conventions.

x = scalar (lower case)

$\vec{x}$ = general vector (lower case, arrow) indexed as $\vec{x}[r]$

$\hat{x}$ = filter estimate vector (lower case, hat)

A = matrix (capital letters) indexed as $A[r, c]$

A^{-1} = matrix inverse

I = the identity matrix

β^{-} = matrix/vector *prediction* (super minus)

β^{T} = matrix/vector transpose (super T)

α_i = matrix/vector/scalar identifier (subscript)

$E\{\bullet\}$ = mathematical expectation

Welch 1997 at Section 3.

3.1 Tracking

3.1.1 Main Tracker Filter

The use of a Kalman filter requires a mathematical (state-space) model for the dynamics of the process to be estimated, the target motion in this case. While several possible dynamic models and associated state configurations are possible, we have found a simple *position-velocity* model to suffice for the dynamics of our applications. In fact we use this same form of model, with different parameters, for all six of the position and orientation components $(x, y, z, \phi, \theta, \psi)$. Discussion of some other potential models and the associated trade-offs can be found in [7] pp. 415-420. Because our implementation is discrete with inter sample time δt we model the target's dynamic motion with the following linear difference equation:

$$\vec{x}(t+\delta t) = A(\delta t)\vec{x}(t) + \vec{w}(\delta t). \qquad (2)$$

Welch 1997 at Section 3.1.

Exhibit E-7

CLAIM 6	Welch 1997
	In the standard model corresponding to equation (2), the n dimensional Kalman filter *state vector* $\hat{x}(t)$ would completely describe the target position and orientation at any time t. In practice we use a method similar to [2,6] and maintain the complete target orientation externally to the Kalman filter in order to avoid the nonlinearities associated with orientation computations. In the internal state vector $\hat{x}(t)$ we maintain the target position as the Cartesian coordinates (x, y, z), and the *incremental* orientation as small rotations (ϕ, θ, ψ) about the (x, y, z) axis. Externally we maintain the target orientation as the *external quaternion* $\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z))$. (See [9] for discussion of quaternions.) At each filter update step, the incremental orientations (ϕ, θ, ψ) are factored into the external quaternion $\hat{\alpha}$, and then zeroed as shown below. Thus the incremental orientations are linearized for the EKF, centered about zero. We maintain the derivatives of the target position and orientation internally, in the state vector $\hat{x}(t)$. We maintain the angular velocities internally because the angular velocities behave like orthogonal vectors and do not exhibit the nonlinearities of the angles themselves. The target state is then represented by the $n = 12$ element internal state vector $$\hat{x} = \left[x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \phi\ \theta\ \psi\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}\right]^T \quad (3)$$ and the four-element external orientation quaternion $$\hat{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z)), \quad (4)$$ where the time designations have been omitted for clarity. Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 6	Welch 1997

The $n \times n$ *state transition matrix* $A(\delta t)$ in (2) projects the state forward from time t to time $t + \delta t$. For our linear model, the matrix implements the relationships

$$\begin{aligned} x(t+\delta t) &= x(t) + \dot{x}(t)\delta t \\ \dot{x}(t+\delta t) &= \dot{x}(t) \end{aligned} \qquad (5)$$

and likewise for the remaining elements of (3).

The $n \times 1$ *process noise vector* $\vec{w}(\delta t)$ in (2) is a normally-distributed zero-mean sequence that represents the uncertainty in the target state over any time interval δt. The corresponding $n \times n$ *process noise covariance matrix* is given by

$$E\{\vec{w}(\delta t)\vec{w}^T(\delta t + \varepsilon)\} = \begin{cases} Q(\delta t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \qquad (6)$$

Because our implementation is discrete with inter sample time δt, we can use the transfer function method illustrated by [7] pp. 221-222 to compute a *sampled* process noise covariance matrix. (Because the associated random processes are presumed to be time stationary, we present the process noise covariance matrix as a function of the inter-sample duration δt only.) The non-zero elements of $Q(\delta t)$ are given by

$$\begin{aligned} Q(\delta t)[i,i] &= \vec{\eta}[i]\frac{(\delta t)^3}{3} \\ Q(\delta t)[i,j] = Q(\delta t)[j,i] &= \vec{\eta}[i]\frac{(\delta t)^2}{2} \\ Q(\delta t)[j,j] &= \vec{\eta}[i](\delta t) \end{aligned} \qquad (7)$$

for each pair

$$(i,j) \in \{(x,\dot{x}), (y,\dot{y}), (z,\dot{z}), (\phi,\dot{\phi}), (\theta,\dot{\theta}), (\psi,\dot{\psi})\}.$$

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 6	Welch 1997

The $\hat{\eta}[i]$ in (7) are the *correlation kernels* of the (assumed constant) noise sources presumed to be driving the dynamic model. We determined a set of values using Powell's method, and then used these in both simulation and our real implementation. The values can be "tuned" for different dynamics, though we have found that the tracker works well over a broad range of values.

The use of a Kalman filter requires not only a dynamic model as described above, but also a *measurement model* for each available type of measurement. The measurement model is used to predict the ideal noise-free response of each sensor and source pair, given the filter's current estimate of the target state as in equations (3) and (4).

It is the nature of the measurement models and indeed the actual sensor measurements that distinguishes a SCAAT Kalman filter from a well-constrained one.

For each sensor *type* σ we define the $m_\sigma \times 1$ *measurement vector* $\hat{z}_\sigma(t)$ and corresponding *measurement function* $\hat{h}_\sigma(\bullet)$ such that

$$\hat{z}_{\sigma,t} = \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t) + \hat{v}_\sigma(t)\,. \qquad (8)$$

Note that in the "purest" SCAAT implementation $m_\sigma = 1$ and the measurements are incorporated as single scalar values. However if it is not possible or necessary to isolate the measurements, e.g. to perform autocalibration, then multi-dimensional measurements can be incorporated also. Guidelines presented in [47] lead to the following heuristic for choosing the SCAAT Kalman filter measurement elements (constraints):

During each SCAAT Kalman filter measurement update one should observe a single sensor and source pair only.

For example, to incorporate magnetic tracker data as an end-user, $m_\sigma = 7$ for the three position and four orientation (quaternion)

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 6	Welch 1997

elements, while if the manufacturer were to use the SCAAT implementation, $m_\sigma = 3$ for each 3-axis electromagnetic response to a single excitation. For an image-based landmark tracker such as [41] the measurement function would, given estimates of the camera pose and a single landmark location, transform the landmark into camera space and then project it onto the camera image plane. In this case $m_\sigma = 2$ for the 2D image coordinates of the landmark.

The $m_\sigma \times 1$ *measurement noise vector* $\hat{\nu}_\sigma(t)$ in (8) is a normally-distributed zero-mean sequence that represents any random error (e.g. electrical noise) in the measurement. This parameter can be determined from component design specifications, and (or) confirmed by off-line measurement. For our simulations we did both. The corresponding $m_\sigma \times m_\sigma$ *measurement noise covariance matrix* is given by

$$E\{\hat{\nu}_\sigma(t)\hat{\nu}_\sigma^T(t+\varepsilon)\} = \begin{cases} R_\sigma(t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \qquad (9)$$

For each measurement function $\hat{h}_\sigma(\bullet)$ we determine the corresponding Jacobian function

$$H_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t)[i,j] \equiv \frac{\partial}{\partial \hat{x}[j]} \hat{h}_\sigma(\hat{x}(t), \hat{b}_t, \hat{c}_t)[i], \qquad (10)$$

where $1 \le i \le m_\sigma$ and $1 \le j \le n$. Finally, we note the use of the standard (Kalman filter) $n \times n$ *error covariance matrix* $P(t)$ which maintains the covariance of the error in the estimated state.

Welch 1997 at Section 3.1.1.

Exhibit E-7

CLAIM 6	Welch 1997

3.1.2 Tracking Algorithm

Given an initial state estimate $\hat{x}(0)$ and error covariance estimate $P(0)$, the SCAAT algorithm proceeds similarly to a conventional EKF, cycling through the following steps whenever a discrete measurement $\vec{z}_{\sigma,t}$ from some sensor (type σ) and source becomes available at time t:

a. Compute the time δt since the previous estimate.

b. Predict the state and error covariance.

$$\begin{aligned} \hat{x}^- &= A(\delta t)\hat{x}(t-\delta t) \\ P^- &= A(\delta t)P(t-\delta t)A^T(\delta t) + Q(\delta t) \end{aligned} \tag{11}$$

c. Predict the measurement and compute the corresponding Jacobian.

$$\begin{aligned} \hat{z} &= \vec{h}_\sigma(\hat{x}^-, \vec{b}_t, \vec{c}_t) \\ H &= H_\sigma(\hat{x}^-, \vec{b}_t, \vec{e}_t) \end{aligned} \tag{12}$$

d. Compute the *Kalman gain*.

$$K = P^-H^T(HP^-H^T + R_\sigma(t))^{-1} \tag{13}$$

e. Compute the *residual* between the actual sensor measurement $\vec{z}_{\sigma,t}$ and the predicted measurement from (12).

$$\vec{\Delta z} = \vec{z}_{\sigma,t} - \hat{z} \tag{14}$$

f. Correct the predicted tracker state estimate and error covariance from (11).

$$\begin{aligned} \hat{x}(t) &= \hat{x}^- + K\vec{\Delta z} \\ P(t) &= (I-KH)P^- \end{aligned} \tag{15}$$

Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 6	Welch 1997
	g. Update the external orientation of equation (4) per the change indicated by the (ϕ, θ, ψ) elements of the state. $\Delta\hat{\alpha} = \text{quaternion}(\hat{x}[\phi], \hat{x}[\theta], \hat{x}[\psi])$ $\hat{\alpha} = \hat{\alpha} \otimes \Delta\hat{\alpha}$ (16) h. Zero the orientation elements of the state vector. $\hat{x}[\phi] = \hat{x}[\theta] = \hat{x}[\psi] = 0$ (17) The equations (11)-(17) may seem computationally complex, however they can be performed quite efficiently. The computations can be optimized to eliminate operations on matrix and vector elements that are known to be zero. For example, the elements of the Jacobian H in (12) that correspond to the velocities in the state $\hat{x}(t)$ will always be zero. In addition, the matrix inverted in the computation of K in (13) is of rank m_σ (2×2 for our example in section 3.4) which is smaller for a SCAAT filter than for a corresponding conventional EKF implementation. Finally, the increased data rate allows the use of the *small angle approximations* $\sin(\theta) = \theta$ and $\cos(\theta) = 1$ in $\check{h}_\sigma(\bullet)$ and $H_\sigma(\bullet)$. The total *per estimate* computation time can therefore actually be less than that of a corresponding conventional implementation. (We are able to execute the SCAAT filter computations, with the autocalibration computations discussed in the next section, in approximately 100µs on a 200 MHz PC-compatible computer.) Welch 1997 at Section 3.1.2.

Exhibit E-7

CLAIM 6	Welch 1997

3.2 Autocalibration

The method we use for autocalibration involves *augmenting* the *main tracker filter* presented in section 3.1 to effectively implement a distinct *device filter*, a Kalman filter, for each source or sensor to be calibrated. (We use the word "device" here to include for example scene landmarks which can be thought of as passive sources, and cameras which are indeed sensors.) In general, any constant device-related parameters used by a measurement function $\hat{h}_\sigma(\bullet)$ from (8) are candidates for this autocalibration method. We assume that the parameters to be estimated are contained in the device parameter vectors $\hat{b}_t$ and $\hat{c}_t$, and we also present the case where both the source and sensor are to be calibrated since omission of one or the other is trivial. We note the following new convention.

$\widehat{\alpha}$ = augmented matrix/vector (wide hat)

Welch 1997 at Section 3.2.

3.2.1 Device Filters

For each device (source, sensor, landmark, etc.) we create a distinct device filter as follows. Let $\hat{\pi}$ represent the corresponding device parameter vector and $n_\pi = \text{length}(\hat{\pi})$.

a. Allocate an $n_\pi \times 1$ state vector $\hat{x}_\pi$ for the device, initialize with the best *a priori* device parameter estimates, e.g. from design.

b. Allocate an $n_\pi \times n_\pi$ noise covariance matrix $Q_\pi(\delta t)$, initialize with the expected parameter variances.

c. Allocate an $n_\pi \times n_\pi$ error covariance matrix $P_\pi(t)$, initialize to indicate the level of confidence in the *a priori* device parameter estimates from (a) above.

Welch 1997 at Section 3.2.1.

Exhibit E-7

CLAIM 6	Welch 1997

3.2.2 Revised Tracking Algorithm

The algorithm for tracking with concurrent autocalibration is the same as that presented in section 3.1, with the following exceptions. After step (a) in the original algorithm, we form augmented versions of the state vector

$$\widehat{x}(t-\delta t) = \left[\hat{x}^T(t-\delta t)\ \hat{x}_{b,t}^T(t-\delta t)\ \hat{x}_{c,t}^T(t-\delta t)\right]^T, \quad (18)$$

the error covariance matrix

$$\widehat{P}(t-\delta t) = \begin{bmatrix} P(t-\delta t) & 0 & 0 \\ 0 & P_{b,t}(t-\delta t) & 0 \\ 0 & 0 & P_{c,t}(t-\delta t) \end{bmatrix}, \quad (19)$$

the state transition matrix

$$\widehat{A}(\delta t) = \begin{bmatrix} A(\delta t) & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}, \quad (20)$$

and the process noise matrix

$$\widehat{Q}(\delta t) = \begin{bmatrix} Q(\delta t) & 0 & 0 \\ 0 & Q_{b,t}(\delta t) & 0 \\ 0 & 0 & Q_{c,t}(\delta t) \end{bmatrix}. \quad (21)$$

Welch 1997 at Section 3.2.2.

Exhibit E-7

CLAIM 6	Welch 1997
	We then follow steps (b)-(h) from the original algorithm, making the appropriate substitutions of (18)-(21), and noting that the measurement and Jacobian functions used in step (c) have become $\check{h}_\sigma(\widehat{x}(t))$ and $H_\sigma(\widehat{x}(t))$ because the estimates of parameters $\check{b}_t$ and $\check{c}_t$ ($\hat{x}_{b,t}$ and $\hat{x}_{c,t}$) are now contained in the augmented state vector $\widehat{x}$ per (18). After step (h) we finish by extracting and saving the device filter portions of the augmented state vector and error covariance matrix $\hat{x}_{b,t}(t) = \widehat{x}(t)[i\ldots j]$ $P_{b,t}(t) = \widehat{P}(t)[i\ldots j, i\ldots j]$ (22) $\hat{x}_{c,t}(t) = \widehat{x}(t)[k\ldots l]$ $P_{c,t}(t) = \widehat{P}(t)[k\ldots l, k\ldots l]$ where $i = n+1$ $j = n+n_b$ $k = n+n_b+1$ $l = n+n_b+n_c$ Welch 1997 at Section 3.2.2.

Exhibit E-7

CLAIM 6	Welch 1997

and n, n_b, and n_c are the dimensions of the state vectors for the main tracker filter, the source filter, and the sensor filter (respectively). We leave the main tracker filter state vector and error covariance matrix in their augmented counterparts, while we swap the device filter components in and out with each estimate. The result is that individual device filters are updated less frequently than the main tracker filter. The more a device is used, the more it is calibrated. If a device is never used, it is never calibrated.

With respect to added time complexity, the computations can again be optimized to eliminate operations on matrix and vector elements that are known to be zero: those places mentioned in section 3.1, and see (19)-(21). Also note that the size of and thus time for the matrix inversion in (13) has not changed. With respect to added space complexity, the autocalibration method requires storing a separate state vector and covariance matrix for each device—a fixed amount of (generally small) space per device. For example, consider autocalibrating the beacon (LED) positions for an optical tracking system with 3,000 beacons. For each beacon one would need 3 words for the beacon state (its 3D position), $3 \times 3 = 9$ words for the noise covariance matrix, and $3 \times 3 = 9$ words for the error covariance matrix. Assuming 8 bytes per word, this is only $3{,}000 \times 8 \times (3 + 9 + 9) = 504{,}000$ bytes.

Welch 1997 at Section 3.2.2.

3.2.3 Discussion

The ability to simultaneously estimate two dependent sets of unknowns (the target and device states) is made possible by several factors. First, the dynamics of the two sets are very different as would be reflected in the process noise matrices. We assume the target is undergoing some random (constant) acceleration, reflected in the noise parameter η of $Q(\delta t)$ in (6). Conversely, we assume the device parameters are constant, and so the elements of $Q_\pi(\delta t)$ for a source or sensor simply reflect any allowed variances in the corresponding parameters: usually zero or extremely small. In addition, while the target is expected to be moving, the filter expects the motion between any two estimations to closely correspond to the velocity estimates in the state (3). If the tracker estimate rate is high enough, poorly estimated device parameters will result in what appears to be almost instantaneous target motion. The increased rate of the SCAAT method allows such motion to be recognized as unlikely, and attributed to poorly estimated device parameters.

Welch 1997 at Section 3.2.3.

CLAIM 6	Welch 1997
	3.3 Stability Because the SCAAT method uses individual measurements with insufficient information, one might be concerned about the potential for instability or divergence. A linear system is said to be stable if its response to any input tends to a finite steady value after the input is removed [24]. For the Kalman filter in general this is certainly not a new concern, and there are standard requirements and corresponding tests that ensure or detect stability (see [18], p. 132): a. The filter must be uniformly completely observable, b. the dynamic and measurement noise matrices in equations (6) and (9) must be bounded from above and below, and c. the dynamic behavior represented by in equation (2) must be bounded from above. As it turns out, these conditions and their standard tests are equally applicable to a SCAAT implementation. For the SCAAT method the conditions mean that the user dynamics between estimates must be bounded, the measurement noise must be bounded, one must incorporate a sufficient set of non-degenerate constraints over time. In particular, the constraints must be incorporated in less than 1/2 the time of the user motion time-constant in order to avoid tracking an alias of the true motion. In general these conditions are easily met for systems and circumstances that would otherwise be stable with a multiple-constraint implementation. A complete stability analysis is beyond the scope of this paper, and is presented in [47]. Welch 1997 at Section 3.3. [47] Greg Welch, 1996. “SCAAT: Incremental Tracking with Incomplete Information,” University of North Carolina at Chapel Hill, doctoral dissertation, TR 96-051. Welch 1997 at References.

CLAIM 6	Welch 1997
	4.1 Tracker Filter The 12 element state vector $\hat{x}(t)$ for the main tracker filter contained the elements shown in (3). Each of the 3000 beacon filters was allocated a 3 element state vector $$\hat{x}_b = \left[x_b\ y_b\ z_b\right]^T$$ where (x_b, y_b, z_b) represents the beacon's estimated position in cartesian (world) coordinates. The 12×12 state transition matrix for the main tracker filter was formed as discussed section 3.1, and for each beacon filter it was the 3×3 identity matrix. The 12×12 process noise matrix for the main tracker was computed using (7), using elements of $\hat{\eta}$ that were determined off-line using Powell's method and a variety of real motion data. For each beacon filter we used an identical noise covariance matrix $$Q_b(\delta t)[i,j] = \begin{cases} \eta_b & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$ for $1 \le i, j \le 3$, with beacon position variance η_b also determined off-line. (See [47] for the complete details.) At each estimate step, the *augmented* 15 element state vector, 15×15 process noise matrix, 15×15 state transition matrix, and 15×15 error covariance matrix all resembled (18)-(21) (without the camera parameter components). The measurement noise model was distance dependent (beacon light falls-off with distance) so $R_\sigma(t)$ from (9) was computed prior to step (d), by using a beacon distance estimate (obtained from the user and beacon positions in the predicted state $\hat{x}^-$) to project a distance-dependent electrical variance onto the camera. Welch 1997 at Section 4.1. *See* Disclosures with respect to Claim 1, *supra*; *see also* Defendants' Invalidity Contentions for further discussion.

F. DEPENDENT CLAIM 8

CLAIM 8	Welch 1997
[8] The method of claim 6 wherein the set of	At least under Plaintiffs' apparent infringement theory, Welch 1997 discloses, either expressly or inherently, the method of claim 6 wherein the set of sensing elements comprises at least one sensor and at least one target, the

CLAIM 8	Welch 1997
sensing elements comprises at least one sensor and at least one target, the sensor making a measurement with respect to the target.	sensor making a measurement with respect to the target. In the alternative, this element would be obvious over Welch 1997 in light of the other references disclosed in Defendants' Invalidity Contentions and/or the knowledge of one of ordinary skill in the art. *See, e.g.*: **1.2 Landmark Tracking** Consider for example a system in which a single camera is used to observe known scene points to determine the camera position and orientation. In this case, the constraints provided by the observations are multi-dimensional: 2D image coordinates of 3D scene points. Given the internal camera parameters, a set of four known coplanar scene points, and the corresponding image coordinates, the camera position and orientation can be uniquely determined in closed-form [16]. In other words if $N = C = 4$ constraints (2D image points) are used to estimate the camera position and orientation, the system is completely observable. On the other hand, if $N < C$ then there are multiple solutions. For example with only $N = 3$ non-collinear points, there are up to 4 solutions. Even worse, with $N = 2$ or $N = 1$ points, there are infinite combinations of position and orientation that could result in the same camera images. Welch 1997 at Section 1.2.

Exhibit E-7

CLAIM 8	Welch 1997
	In general, for closed-form tracking approaches, a well or over-constrained system with $N \geq C$ is observable, an under-constrained system with $N < C$ is not. Therefore, if the individual observations provide only partial information, i.e. the measurements provide insufficient constraints, then multiple devices or landmarks must be excited and (or) sensed prior to estimating a solution. Sometimes the necessary observations can be obtained simultaneously, and sometimes they can not. Magnetic trackers such as those made by Polhemus and Ascension perform three *sequential* source excitations, each in conjunction with a complete sensor unit observation. And while a camera can indeed observe multiple landmarks simultaneously in a single image, the image processing to identify and locate the individual landmarks must be done sequentially for a single CPU system. If the landmarks can move independently over time, for example if they are artificial marks placed on the skin of an ultrasound patient for the purpose of landmark-based tracking [41], batch processing of the landmarks can reduce the effectiveness of the system. A SCAAT implementation might grab an image, extract a *single* landmark, update the estimates of both the camera *and* landmark positions, and then throw-away the image. In this way estimates are generated faster and with the most recent landmark configurations. Welch 1997 at Section 1.2.

Exhibit E-7

CLAIM 8	Welch 1997
	 Figure 3: The HiBall is shown here with the internal circuitry exposed and the lenses removed. The sensors, which can be seen through the lens openings, are mounted on PC boards that fold-up into the HiBall upon assembly. The mechanical pencil at the bottom conveys an indication of the relative size of the unit. Welch 1997 at Figure 3.

Exhibit E-7

CLAIM 8	Welch 1997
	4 EXPERIMENTS We are using the SCAAT approach in the current version of the UNC wide-area optoelectronic tracking system known as the *HiBall tracker*. The *HiBall*, shown below in Figure 3, incorporates six optical sensors and six lenses with infrared filters into one golf ball sized sensing unit that can be worn on a user's head or hand. The principal mechanical component of the HiBall, the senor housing unit, was fabricated by researchers at the University of Utah using their α1 modeling environment. Because the HiBall sensors and lenses share a common transparent space in the center of the housing, a single sensor can actually sense light through more than one lens. By making use of all of these *views* we end up with effectively 26 "cameras". These cameras are then used to observe ceiling-mounted light-emitting diodes (LEDs) to track the position and orientation of the HiBall. This inside-looking-out approach was first used with the previous UNC optoelectronic tracking system [44] which spanned most of the user's head and weighed approximately ten pounds, not including a backpack containing some electronics. In contrast, the HiBall sensing unit is the size of a golf ball and weighs only five ounces, *including* the electronics. The combination of reduced weight, smaller packaging, and the new SCAAT algorithm results in a very ergonomic, fast, and accurate system. In this section we present results from both simulations performed during the design and development of the HiBall, and Welch 1997 at Section 4.

Exhibit E-7

CLAIM 8	Welch 1997

preliminary results from the actual implementation. The simulations are useful because we have control over the "truth" and can perform controlled experiments. The results from the actual implementation serve to demonstrate actual operation and to provide some support for our accuracy and stability claims.

With respect to the SCAAT implementation, the tracker *sensors* are the HiBall cameras and the tracker *sources* are the ceiling-mounted 2D array of approximately 3000 electronic beacons (LEDs). The cameras provide a single 2D measurement vector, i.e. a 2D constraint, which is the (u, v) image coordinates of the beacon as seen by the camera. So for this example, $m_\sigma = 2$ and $z_\sigma = [u, v]^T$. The measurement function $h_\sigma(\bullet)$ transforms the beacon into camera coordinates and then projects it onto the camera's image plane in order to predict the camera response.

For the simulations we generated individual measurement events (a single beacon activation followed by a single camera reading) at a rate of 1000 Hz, and corrupted the measurements using the noise models detailed in [8]. We tested components of our real system in a laboratory and found the noise models in [8] to be reasonably accurate, if not pessimistic. We also perturbed the 3D beacon positions prior to simulations with a normally-distributed noise source with approximately 1.7 millimeters standard deviation. We controlled all random number generators to facilitate method comparisons with common random sequences.

Welch 1997 at Section 4.

To evaluate the filter performance we needed some reference data. Our solution was to collect motion data from *real-user* sessions with a conventional tracking system, and then to filter the data to remove high frequency noise. We then *defined* this data to be the "truth". We did this for seven such sessions.

The simulator operated by sampling the truth data, choosing one beacon and camera (round-robin from within the set of valid combinations), computing the corresponding camera measurement vector $\hat{z}_{\sigma,t}$, and then adding some measurement noise. The (noisy) measurement vector, the camera parameter vector $\hat{c}_t$ (position and orientation in user coordinates), and the beacon parameter vector $\hat{b}_t$ (position in world coordinates) were then sent to the tracker.

Welch 1997 at Section 4.

CLAIM 8	Welch 1997
	For the tracking algorithm, we simulated both the SCAAT method (section 3.1, modified per section 3.2 for autocalibration) and several multiple-constraint methods, including the Collinearity method [3] and several variations of moving window (finite impulse response) methods. For each of the methods we varied the measurement noise, the measurement frequency, and the beacon position error. For the multiple constraint methods we also varied the number of constraints (beacon observations) per estimate N. In each case the respective estimates were compared with the truth data set for performance evaluation. Welch 1997 at Section 4. *See* Disclosures with respect to Claim 6, *supra*; *see also* Defendants' Invalidity Contentions for further discussion.

G. DEPENDENT CLAIM 9

CLAIM 9	Welch 1997
[9] The method of claim 8 wherein the target comprises a natural feature in an environment.	At least under Plaintiffs' apparent infringement theory, Welch 1997 discloses, either expressly or inherently, the method of claim 8 wherein the target comprises a natural feature in an environment. In the alternative, this element would be obvious over Welch 1997 in light of the other references disclosed in Defendants' Invalidity Contentions and/or the knowledge of one of ordinary skill in the art. *See, e.g.*:

Exhibit E-7

CLAIM 9	Welch 1997

1 INTRODUCTION

The method we present requires, we believe, a fundamental change in the way people think about estimating a set of unknowns in general, and tracking for virtual environments in particular. Most of us have the preconceived notion that to estimate a set of unknowns we need as many constraints as there are degrees of freedom at any particular instant in time. What we present instead is a method to constrain the unknowns *over time*, continually refining an estimate for the solution, a *single constraint at a time*.

For applications in which the constraints are provided by real-time observations of physical devices, e.g. through measurements of sensors or visual sightings of landmarks, the SCAAT method isolates the effects of error in individual measurements. This isolation can provide improved filtering as well as the ability to individually calibrate the respective devices or landmarks concurrently and continually while tracking. The method facilitates user motion prediction, multisensor or multiple modality data fusion, and in systems where the constraints can only be determined sequentially, it provides estimates at a higher rate and with lower latency than multiple-constraint (batch) approaches.

Welch 1997 at Section 1.

1.2 Landmark Tracking

Consider for example a system in which a single camera is used to observe known scene points to determine the camera position and orientation. In this case, the constraints provided by the observations are multi-dimensional: 2D image coordinates of 3D scene points. Given the internal camera parameters, a set of four known coplanar scene points, and the corresponding image coordinates, the camera position and orientation can be uniquely determined in closed-form [16]. In other words if $N = C = 4$ constraints (2D image points) are used to estimate the camera position and orientation, the system is completely observable. On the other hand, if $N < C$ then there are multiple solutions. For example with only $N = 3$ non-collinear points, there are up to 4 solutions. Even worse, with $N = 2$ or $N = 1$ points, there are infinite combinations of position and orientation that could result in the same camera images.

Welch 1997 at Section 1.2.

Exhibit E-7

CLAIM 9	Welch 1997
	In general, for closed-form tracking approaches, a well or over-constrained system with $N \geq C$ is observable, an under-constrained system with $N < C$ is not. Therefore, if the individual observations provide only partial information, i.e. the measurements provide insufficient constraints, then multiple devices or landmarks must be excited and (or) sensed prior to estimating a solution. Sometimes the necessary observations can be obtained simultaneously, and sometimes they can not. Magnetic trackers such as those made by Polhemus and Ascension perform three *sequential* source excitations, each in conjunction with a complete sensor unit observation. And while a camera can indeed observe multiple landmarks simultaneously in a single image, the image processing to identify and locate the individual landmarks must be done sequentially for a single CPU system. If the landmarks can move independently over time, for example if they are artificial marks placed on the skin of an ultrasound patient for the purpose of landmark-based tracking [41], batch processing of the landmarks can reduce the effectiveness of the system. A SCAAT implementation might grab an image, extract a *single* landmark, update the estimates of both the camera *and* landmark positions, and then throw-away the image. In this way estimates are generated faster and with the most recent landmark configurations. Welch 1997 at Section 1.2. *See* Disclosures with respect to Claim 8, *supra*; *see also* Defendants' Invalidity Contentions for further discussion.